2004-03

Toward managing & automating CyberCIEGE scenario definition file creation

Johns, Kenneth W., Jr.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

TOWARD MANAGING & AUTOMATING CYBERCIEGE SCENARIO DEFINITION FILE CREATION

by

Kenneth W. Johns, Jr.

March 2004

Thesis Co-Advisor:	Cynthia Irvine
Thesis Co-Advisor:	Paul Clark
Second Reader:	Mike Thompson

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** March 2004	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Toward Managing & Automating CyberCIEGE Scenario Definition File Creation			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kenneth W. Johns, Jr.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The CyberCIEGE project seeks to create an alternative to traditional Information Assurance (IA) training and education approaches by developing an interactive, entertaining commercial-grade PC-based computer game/virtual laboratory. CyberCIEGE will provide a robust, flexible and extensible gaming environment where each instance of the game is based on a fully customizable *scenario*. These *scenarios* are written in the CyberCIEGE *Scenario Definition Language*. Unfortunately, the trade-off for flexibility, extensibility and fully customizable *scenarios* is syntax complexity in the *scenario definition language*.

This thesis will solve this real world problem by showing that the complexity of *scenario definition language* syntax can be managed through a software tool. This thesis will develop such a tool and further demonstrate that progress can be made toward automating *scenario* generation.

14. SUBJECT TERMS Information Assurance, CyberCIEGE, Syntax Complexity Management			**15. NUMBER OF PAGES** 484
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

TOWARD MANAGING & AUTOMATING CYBERCIEGE SCENARIO DEFINITION FILE CREATION

Kenneth W. Johns, Jr.
Civilian, Federal Cyber Service Corp, Naval Postgraduate School
B.S., California State University Bakersfield, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2004

Author: Kenneth W. Johns, Jr.

Approved by:
Cynthia Irvine
Thesis Co-Advisor

Paul Clark
Thesis Co-Advisor

Mike Thompson
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The CyberCIEGE project seeks to create an alternative to traditional Information Assurance (IA) training and education approaches by developing an interactive, entertaining commercial-grade PC-based computer game/virtual laboratory. CyberCIEGE will provide a robust, flexible and extensible gaming environment where each instance of the game is based on a fully customizable *scenario*. These *scenarios* are written in the CyberCIEGE *Scenario Definition Language*. Unfortunately, the trade-off for flexibility, extensibility and fully customizable *scenarios* is syntax complexity in the *scenario definition language*.

This thesis will solve this real world problem by showing that the complexity of *scenario definition language* syntax can be managed through a software tool. This thesis will develop such a tool and further demonstrate that progress can be made toward automating *scenario* generation.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The successful completion of this thesis would not have been possible were it not for the support of many individuals.

I would first like to thank my wife and children for their love and support. Thank you for tolerating my many absences from our family.

I would like to thank Dr. Cynthia Irvine, Paul Clark and Mike Thompson, my advisory team. Your insight, guidance, comments and support have been invaluable in completing this thesis and have made it what it is today.

To Al Wong and Jennifer Guild, thank you for helping me find my way though Java when I was most lost.

To Tanya Raven, Paul Schoberg, Danny Craven, Jason Cisneros, John Clark, Randy Christensen, Scott Cote, JD Fulp, Daniel Warren and Jim & Jennifer Guild. You are all good friends, thank you for helping me maintain my sanity and stay the course.

This material is based upon work supported by the National Science Foundation under Grant No. DUE-0210762. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION & BACKGROUND

A. THESIS STATEMENT

The purpose of this thesis is to ask and answer two questions. Can a software tool be developed to manage the complexity of CyberCIEGE scenario language syntax? Can reusable CyberCIEGE scenario objects make scenario creation more automated? By answering these questions, this thesis will simplify the generation of syntactically correct CyberCIEGE scenario generation and will help advance the goals of the CyberCIEGE project.

B. THESIS SCOPE & LAYOUT

This thesis describes the development of a version 1.0 CyberCIEGE Scenario Definition Tool used to generate scenario definition files based on the October 1, 2003 version of the scenario language.

The thesis chapters will be laid out in the following order:

- Chapter I - Introduction & Background – This chapter will introduce the project and provide background discussion on the motivation behind this thesis.
- Chapter II – Requirements Analysis – This chapter describes the requirements of a scenario definition tool that will answer the questions posed in the thesis statement.
- Chapter III – Design & Implementation – This chapter will discuss how the scenario definition tool was designed and implemented.
- Chapter IV - Testing – The testing methodology will be discussed.
- Chapter V - Future Work & Conclusion – This chapter will suggest improvements to the scenario definition tool, and then conclude with some final thoughts on the project and its success.

C. THE NEED FOR AN ALTERNATIVE

The problem with computer security training and education today is that it can be, in a word, boring [Irvine1 2003]. It is boring for the employee who is forced to sit in a conference room and listen to the security manager talk about policies and why passwords have to be a certain length, and why they have to log off of their workstations when they are away, etc. It is boring for the student who has to sit in a classroom and listen to lectures on the subtle nuances of security policies and then try to understand

those concepts without any way to experience those subtleties first hand. It is boring (and not terribly fulfilling) for the instructors and administrators of those meetings who have worked hard to assemble their information and present it to a, less than enthusiastic, captive audience.

In addition to not always being interesting, computer security can be difficult to comprehend. It is not reasonable to expect that the average computer user will understand fail secure/fail safe, the principle of least privilege [Saltzer 1975] and similar concepts. Most users do not even appreciate why they need to have long and complex passwords. Moreover, security policy can be discussed but it is not practical or feasible to build an entire network that implements that policy for the sole purpose of giving students hands on experience. It is dangerous and costly to leave systems up and insecure for the sole purpose of demonstrating the risks of the Internet and open networks [Irvine1 2003]. Policy is subtle, and there is no easy way to demonstrate how one small change in implementation can affect the entire system. The average user of IT has little or no appreciation for security measures and policy. It is critical that all users of IT have an appreciation for computer security.

What if there were a tool that could simulate these and other security issues? What if there was an alternative to traditional training techniques that was engaging and interactive? What if a tool existed that could be used in the classroom and in the board room, a tool that could help everyone from the average user to policy makers to students studying computer security learn what they need to learn and enjoy it?

1. CyberCIEGE: The Virtual Laboratory and Simulation

The description contained in the following sections is derived from [Irvine2 2003]. The Center for Information Systems Security Studies and Research (CISR) at the Naval Postgraduate School (NPS) is in the process of creating such an instructional tool. The tool, called CyberCIEGE, is designed to help solve many of the previously mentioned problems and more. CyberCIEGE represents a partnership between academia and the gaming industry. A professional game development company, Rivermind, is writing the game: its graphics and simulation engine. This will ensure gaming industry

quality in the end product. CISR is responsible for the accuracy of the simulation engines behavior, the academic content and integrity. This will ensure that the game is accurate enough to be used in the classroom/lab and for training and awareness purposes.

CyberCIEGE is a resource management video game that will enable players to implement security policy on a simulated network. Players may take on (to a limited extent) or observe one of several roles to include (but not limited to):

- System administrator
- Manager
- System user

CyberCIEGE is also an information assurance virtual laboratory. CyberCIEGE will allow instructors to develop playable scenarios that students can run in a lab environment in conjunction with course work. Afterwards the instructor can use an add-on tool [Teo 2003] to collect data about student performance to aid in grading. The virtual laboratory is unique in that it adapts to player decisions (to include or omit a piece of hardware, configuration setting etc.) and reports the effects by changing the available, limited, resources and success of the enterprise.

2. Hands On Security Experience

CyberCIEGE will provide players with the ability to apply real world scenarios to abstract policies and concepts they could only read about before. Moreover, players will have a stake in the outcome of the scenario. They will be driven by the will to succeed and win. They will have a real interest in the survival of their virtual network, just as they would if they were defending a real network from attack. The real advantage comes when they lose, as no real world data or systems have been compromised but the pain of the loss is still quite real.

3. Multiple Modes of Operation

CyberCIEGE has three intended modes of operation: stand-alone game, and self-paced tutorial, and as a laboratory combined with a course. As a stand-alone game CyberCIEGE is useful in introducing players to basic IA concepts and terms. In this mode, players will play through a standard set of basic scenarios. As a self-paced tutorial CyberCIEGE can be used in training environments by putting players through a set of scenarios customized to the training environment and goals of the organization

sponsoring the training. Finally, when used as a laboratory tool in conjunction with a course, the game can be played using a few custom scenarios or a series of related scenarios that focus on a central theme.

4. Maximum Flexibility: Scenario Based Learning

Normally a video game or training software ships with a single fixed set of missions/lessons. If additional missions/lessons are desired then the software must be updated or add-on modules must be developed; CyberCIEGE is different. CyberCIEGE can be shipped with a set of starter scenarios or with no scenarios at all. This is possible because of the robust scenario engine that the game is built around. The game is designed to be extensible. This was done so that the game could evolve to simulate new threats and countermeasures [Irvine 2003]. It was also done so that custom scenarios and sets of scenarios could be developed to facilitate teaching specific concepts.

The concept of a scenario in CyberCIEGE is synonymous with a lesson or a mission in any other training program or video game. Where scenarios cease to be like their counterparts is in their ability to be modified quickly and inexpensively with immediate re-playability. Scenarios are based on their own flexible and robust language, which gives scenario designers the freedom to write their own scenarios without any support from the game developer -- not even so much as a recompilation of the game.

D. THE NEED FOR A SCENARIO DEFINITION TOOL

The description contained in the following sections is derived from [Rivermind 2003]. To use an old cliché, “Freedom is not free”. This is certainly true for CyberCIEGE scenarios. The freedom to quickly change existing scenarios and to write new scenarios requires a robust scenario language. With that robustness comes complexity and the scenario language for CyberCIEGE is quite complicated.

The CyberCIEGE engine is able to run any scenario written in the scenario language by reading and parsing an external file called a scenario definition file (SDF). The SDF provides the game engine with designer-specified startup information, win/loss conditions, goals, etc. Additional details are described in a later section. The SDF is based on the scenario format template (SFT).

1. The Scenario Format Template

The SFT is the master document that describes the scenario language and how it is to be used to create scenarios. It specifies the layout of an SDF, in what order fields are expected to appear, what is optional and what is not.

Both CISR representatives and Rivermind representatives created the SFT over the course of several meetings. The goal of the SFT was to create a mechanism for the extensibility called for by the project while maintaining the scientific and academic integrity of the security concepts to be modeled by the game engine. Moreover, the Rivermind representatives had a stake in keeping the SFT realistic enough to write an engine for and to ensure exciting and interactive game play. Several thesis students (the author included) participated in the meetings to provide additional perspective on the SFT and how game play would be affected. All of the student representatives were avid users of video games and could provide insight from that perspective. In addition, the students were less experienced in IA concepts than the more senior CISR staff members were. The disparity in IA experience, between CISR senior staff members and the students, led to discussions and decisions that helped to keep the game playable for users who would come to the game with little or no IA background.

The SFT specifies that a scenario be composed of seventeen descriptors. The descriptors were selected based on the descriptor's role in communicating information to the game engine and user, and the descriptor's relevance to IA concepts and modeling those concepts. The descriptors are presented here in the order defined by the SFT: Organization, Site, Zone, Department, Network, Secrecy, Integrity, DACGroup, Asset, AssetGoal, User, Component, Briefing, DebriefWin, DebriefLose, Conditions and Triggers. Based on the data fields contained within each of these descriptors a custom scenario can be created to teach a specific lesson. Each of the descriptors for a scenario is briefly explained below.

a. Organization

The Organization descriptor defines the organization, its name, type, the start date for the scenario, and the scenario's title, starting funds, budget and the amount of profit that flows back into the IT budget. There is only one organization entry per scenario.

b. Site

The site descriptor provides which floor plan will be used (taken from a fixed set of previously created floor plans provided by the game developer). It also provides a description of the site, city and state and determines if players can change the attributes of components within the site and the site's name. There is only one site entry per scenario.

c. Zone

The Zone descriptor defines the physical locations where people and components can be placed. By providing the upper-left-hand corner and lower-right-hand corner values, scenario designers define the dimensions of a Zone. Zone further provides a set of default procedural and physical security settings that will apply to any component or user placed in this zone. If a component has procedural security settings of its own, they override zone defaults but they are still subject to the physical security settings of the zone. There is a zone entry for each zone.

d. Department

The Department descriptor defines a department for this scenario (e.g., Marketing, Accounting, R&D). There is one department entry for each department.

e. Network

The Network descriptor defines a single network by name and IP (Ipv4) address. The security properties of a network are captured in a component's network connection settings. There is a network entry for each network.

f. Secrecy

The Secrecy descriptor defines a secrecy label to be used in a MAC policy. The description includes enough information for the game engine to simulate the dominance relationship between multiple labels. The descriptor also defines the value of the secret to the enterprise and the value of the secret to attackers. The descriptor also optionally defines the rate of change for the secrecy value to the enterprise and the attacker. Finally, the descriptor defines the initial background check required for users who are cleared to access data at this level of secrecy. There is a secrecy entry for each label to be used in the scenario.

g. Integrity

The Integrity descriptor defines an integrity label to be used in a MAC policy. The description includes enough information for the game engine to simulate the dominance relationship between multiple labels. The descriptor also defines the value of the data for a given level of integrity, the rate of change for the integrity value, how motivated an attacker is to corrupt the data, how much an attackers motivation changes per month and finally the initial background check required for users to access data at this level of integrity. There is an integrity entry for each label to be used in the scenario.

h. DACGroup

The DACGroup descriptor defines all of the DAC groups in the game. There is only one DACGroup entry.

i. Asset

The Asset descriptor defines a data asset for a scenario. Asset is either "instantiated" when a scenario starts or not. If the Asset is not "instantiated" then a virtual user, at some point in the game, must create it. The descriptor also defines the secrecy and integrity labels for the Asset, an attackers motive to deny access, the penalty if the Asset is unavailable, and the intended access control list for the Asset and the cost associated with an access violation. There is an asset entry for each asset whether it is instantiated or not.

j. AssetGoal

The AssetGoal descriptor defines the requirement for virtual users to be able to access a specific Asset or Assets. The descriptor provides for shared Asset access where several users must be able to access the same Asset (not necessarily at the same time) and simultaneous access of several different Assets. Finally, the descriptor defines the cost associated with not being able to access the Asset and how that cost changes per month. There are multiple AssetGoal entries per scenario.

k. User

The User descriptor defines two types of virtual users, support staff and users who have goals. Support staff users have a certain amount of skill with an IT system and can be fired by the game player. Support Staff have a date that they become available. Users with goals have secrecy and integrity clearances. Users may belong to

one or more DACGroups. All users have trustworthiness, initial training levels, happiness, productivity values, skill, a monthly cost, a gender and a description. Users with goals have AssetGoals. There is a User entry for each user.

l. Component

Component is a very complex descriptor that captures many things regarding computer systems in the game. There are two types of Components: templates (called a catalog component) and non-templates (called a physical component). If a Component is a template then it exists in the games catalog of Components available for purchase at game time. If a Component is not a template then it is a physical component that is instantiated at game time and may belong to a user.

Some of the more significant data that the descriptor captures for both types of Components includes PKI policies, password policies, software administration, browser, email, patch update and anti virus settings. Physical components have assets, configuration settings, may be protected by filters, have network connections, have single-level or multi-level connections and have default procedural settings. There is a component entry for each catalog and physical component.

m. Briefing

Briefing is a block of text that communicates what the designer wants the player to know at game startup. This may include the state of affairs at the moment the game starts, specific goals, policies to be enforced or existing problems to be solved. There is a single briefing entry per scenario.

n. DebriefWin

DebriefWin is the displayed message if a player wins. There is one DebriefWin entry per scenario.

o. DebriefLose

DebriefLose is the displayed message if a player loses. There is one DebriefLose entry per scenario.

p. Conditions

Conditions are descriptors that determine whether a Trigger will occur.

Conditions evaluate to *true* or *false*. When a Condition or list of Conditions bound to a Trigger is true, the Trigger fires. There is one condition entry for each condition in a scenario.

q. Triggers

Triggers fire based on the state of conditions. Triggers are useful for causing designer-specified events to occur at given times and game state. There are different kinds of Triggers, for instance, win/loss Triggers, message display, change in game state (e.g., give the player more money, cause an attack to occur or not occur). There is one Trigger entry for each Trigger in a scenario.

2. Toward Automating Scenario Design

While it is possible to write scenarios by hand, this is not advisable. A scenario written by hand is subject to typographical errors and omissions. These errors may cause a scenario to fail to run, to run with some descriptors being ignored or to run in a manner that is completely unexpected. SFT syntax is complicated and it is a long and tedious process to create even a very small and simple scenario by hand. The advantage of SFT syntax is that it (like any good language) is repetitive and follows a set of well-defined rules. Moreover, it should not be a requirement that all scenario designers know SFT syntax in order to write scenarios. It is assumed that all scenario designers will have advanced knowledge of IA concepts especially those concepts that they want their scenario to teach. It should not be a requirement that they have working knowledge of SFT syntax. To that end, it has been proposed that SFT syntax lends itself to being abstracted into an easy to use GUI-based application.

By abstracting SFT syntax into GUI-based forms software, will now be able to track all of the information provided, test for critical omissions and quickly, easily and accurately produce a properly formatted SDF with minimal or no syntax errors. It is important to point out that all scenarios are subject to logic errors on the part of the designer and that such an application would not be able to prevent such errors. In addition, instead of a scenario designer having to repeatedly generate commonly used descriptors that capture a lot of information (e.g. user), the software would store the designer's decisions for several instances of the same descriptor and then quickly write all of the saved instances to a file.

This scenario definition tool (SDT) would serve three purposes. First, it would aid in the creation of syntactically correct SDFs (as mentioned above). Second, it would provide a way to create and store reusable descriptors. These reusable sets of descriptors could be used to build scenarios and would drastically cut down on the amount of time required to build a series of related scenarios. Furthermore, the SDT would be able to save master copies of scenarios for later editing, reuse and for the creation of another series of scenarios. Third, it would begin the process of automating SDF creation. It is unlikely that SDF creation will ever be fully automated but this tool would go a long way to making the process easier and faster. It stands to reason that the easier it is to create an SDF, then more scenarios will be created.

In the next chapter, the requirements for a scenario definition tool are identified and discussed.

II. REQUIREMENTS ANALYSIS

A. INTRODUCTION

Plato said, "The beginning is the most important part of the work" and the collection of requirements for a software application is no exception. The beginning of requirements collection and analysis for the Scenario Definition Tool started with the observation that Scenario Format Template (SFT) language syntax is complex and cumbersome. This observation about SFT syntax has lead to the two thesis questions that ultimately drove this project. One, can a software tool be developed to manage the complexity of SFT syntax? Two, Can reusable scenario descriptor sets support automation of scenario creation?

1. Can a Software Tool be Developed to Manage the Complexity of SFT Syntax?

If the answer to this question is "yes" then scenario designers will be free of the complexity and tedium of SFT syntax. A balance will be created between mundane, mechanical data input, automation and flexibility. Designers will be free to focus on the content of their scenarios and not the accuracy of their scenario definition files (SDF), as the software will handle that for them. Designers will be more efficient when they develop a new scenario and will be able to generate more scenarios faster. Scenario definition files will be generated in minutes rather than days.

If the answer to this question is "no" then there are a couple of possible outcomes. First, the software has failed to meet its requirements and the complexity of scenario design will have been translated, in part or in whole, directly to the solution and little or nothing will be gained. Scenario designers will still be burdened by complexity but now it will be the complexity of the solution and not the SFT syntax. Second, and much worse, there will be an accumulation of complexity. Some or all of the complexity of SFT syntax will still be evident coupled with a newly created, artificial, complexity imposed by the solution. Scenario generation will continue to be slow and difficult.

2. Can Reusable Scenario Descriptor Sets Support Automation of Scenario Creation?

The idea of reusable scenario descriptors is not inherent to the CyberCIEGE game engine or the SFT; it is made possible by the Scenario Definition Tool. This new capability is significant and worthy of further investigation.

First, the phrase "reusable scenario descriptor" must be defined. In the previous chapter, the concept of SFT descriptors (e.g., Secrecy) was introduced. It is possible to build a scenario by asking for the information required by a descriptor, using that information for the single build and then discarding it. In contrast, after asking for the information required by a descriptor if one were to save that information, the descriptor or some modification of it could be used again later. While descriptor reuse could be performed manually by cutting and pasting desired descriptors to and from various scenarios such an approach to reusability would be tedious and inefficient. A tool would make this task faster and easier. It would be possible to change the content of one reusable descriptor and have that change affect many scenarios. Moreover, the abstract representation of the descriptor would be easier to read and edit. In the next chapter, Design, the reusable descriptors will be identified.

If the answer to the question above is "yes" then it will be possible to create a library of reusable descriptor sets (henceforth called *reusable sets*). A *reusable set* is a collection of descriptors of the same type, all of which belong to the same scenario and are stored together. These *reusable sets* will be easy to manipulate and add to other scenarios. If the information contained in a *reusable set* is the same for several scenarios then hours and perhaps days can be saved in scenario development time. If scenarios can be saved and reused, then a master copy of that scenario can be kept with the SDT for later editing and possible reuse. These reusability features, when taken together, form the means to generate and manage multiple scenarios and automate their generation.

While it may seem intuitive that one would make scenarios and descriptors reusable, this capability is not necessary to generate SDFs. As mentioned earlier, reusable scenario descriptors are not inherent to the SFT but a benefit of building a software tool. A very minimal software tool may avoid the time and trouble of adding

this functionality. However, *reusable sets* will enhance the scenario designers experience with the SDT and increase its usability greatly.

If the answer to the question is "no", then again the software has failed to meet is requirements and reusability will be cumbersome or may actually interfere with scenario creation. Such interference is likely to come in the form of additional complexity and confusion regarding how to use the reusability features. Furthermore, it is possible that a failure of the reusability features could prevent automation of scenario creation, make the software too frustrating to use and ultimately render it irrelevant.

B. ENVIRONMENT

A successful solution would be an advance toward automating CyberCIEGE scenario creation. Furthermore, a successful solution would free scenario designers from having to manage SFT syntax and all of its complexity. Designers would be free to concentrate their full effort on the content of the scenario. The process of scenario creation would be faster and less frustrating. In the best case, a successful solution would make scenario creation so easy that large numbers of diverse, entertaining and very useful scenarios would be created. Since the Scenario Definition Tool is used to support CyberCIEGE, it is expected that it will run in the same environments that CyberCIEGE runs in. CyberCIEGE is designed to run on Windows-based personal computers. As a result, the SDT has been designed to run on that platform as well.

The target user for this software tool is a college-educated professional who is familiar with CyberCIEGE and has advanced knowledge of IA concepts. These users will be very familiar and comfortable with software and computer systems. It is expected that the most common user of this software tool will be professors and lecturers at the undergraduate and graduate levels seeking to enhance their IA lectures with CyberCIEGE labs. Another likely user is a corporate/government/military security professional responsible for security training. Researchers in the area of IA would be another class of users that would find value in a Scenario Definition Tool. Finally, it is possible that a small number of standard users of CyberCIEGE will want to develop some of their own scenarios.

C. CONSTRAINTS

One broad restriction was placed on the SDT. That was, to allow full use of the SFT language without restrictions on the designer, while simplifying the scenario creation process. The main concern was that very robust error checking might cause part of the SFT language capabilities to be lost. With this restriction comes a trade off. By allowing scenario designers great freedom, comes the possibility that an SDF can be generated that the game engine cannot run.

Another constraint placed on the SDT was that it be programmed in Java to run on a Windows platform. Java was chosen because it is a flexible well-documented language that many programmers are familiar with. Moreover, the next person to work on the project and maintain it is likely to be another thesis student and there is a very high likelihood that the student will be familiar with Java (either through prior course work or because Java is taught as the Naval Postgraduate School). The Windows constraint is based on the SDT relationship to CyberCIEGE. CyberCIEGE is a Windows-based game therefore; it was felt that the Scenario Definition Tool should run on the same operating system.

The final constraint placed on the SDT was the amount of time available to design and develop the software.

D. REQUIREMENTS

Good output would come in the form of an SDF formatted according to the standards set in then SFT. The generated scenario file should be free of syntax errors and provide the data necessary to start and successfully run the intended scenario to completion. The tool should also output reusable sets of descriptors and scenario masters. These files should be able to be saved, opened, edited and saved again without corruption or loss of data.

The scenario definition tool should address three problems. One, it should deal with the complexity of SFT language syntax by abstracting it into a GUI based software application that is less complex than the language. The application should eliminate the need for scenario designers to have to learn the scenario language.

Two, it should provide automated support for CyberCIEGE scenario creation by helping to manage scenarios and descriptors. This management should come in the form of storing reusable sets and scenarios for later use. Moreover, the software should provide the means to move between reusable set elements, edit those elements and remove them at will. Later, the designer should be able to quickly and easily add descriptors to a scenario.

Three, it should prevent syntax errors from being introduced into the final scenario definition file.

E. SUMMARY

After the data from several iterations of requirements interviews was analyzed, a list of core requirements was identified:

- GUI based
- Identify reusable descriptors
- Collect the identified descriptors of the same type into sets that are reusable
- Provide the means to navigate between the elements of a set
- Provide the means to add/remove reusable sets to/from scenarios
- Provide the means to manage a scenario in all stages of development
- Provide the means to generate an SDF
- Provide the means to edit scenarios and reusable set elements
- Allow full use of the SFT language without restrictions on the designer. No part of the SFT Language can be lost.
- Efficient performance

This list provided sufficient direction to begin designing the Scenario Definition Tool. The next chapter will discuss this design in detail.

THIS PAGE INTENTIONALLY LEFT BLANK

III. DESIGN & IMPLEMENTATION

A. INTRODUCTION

This chapter will discuss the process and details of the design and implementation of the Scenario Definition Tool. The first part of the chapter will cover the high level design. This discussion will include the definition of reusable sets and the components of the human interface. The discussion presented in the second part of the chapter will move on to lower level design features. The process of converting descriptors to forms will be dealt with, then the relationship between the human interface components and the data structures they represent will be discussed. The second part of the chapter will end with a discussion that relates the data structures to the process of creating a Scenario Definition File. The third part of the chapter will highlight the class level organization of the SDT and will identify in what classes the data structures of the interface reside. This will complete the mapping from interface to data structure to class. The final part of the chapter will deal with the implementation of the SDT. This part of the chapter will briefly discuss the rationale behind the language and IDE used to create the SDT.

B. HIGH LEVEL DESIGN

1. Reusable Sets

The first part of the design process involved determining which descriptors would be reusable (the concept of *reusable sets* was introduced in the previous chapter) and creating a criterion to support those decisions. Two criteria were established. A descriptor was designated reusable when:

- A significant amount of design choices were captured by that descriptor.
- By being reusable, the descriptor could potentially save a lot of time in the scenario design process.

Admittedly these criteria are subjective, therefore, the following discussion will illustrate some of the thought process that went into designating reusable descriptors. Some descriptors, *User* for instance, captures a lot of information that takes a considerable amount of time to input. If a scenario designer had to make these design choices about *User* every time he or she wrote a new scenario (especially if the scenario was part of a series of related scenarios) it would be time consuming and frustrating.

Of the seventeen original descriptors, thirteen were identified as being reusable. One of the descriptors, Component, which captures an enormous amount of information, was split into two different reusable descriptors (Catalog Component and Physical Component) yielding fourteen reusable descriptors. Component was broken up because components can appear in the game as items in a catalog available for purchase (in scenario language they are called templates) and as computer systems that belong to virtual users (they are "physical components" in the game world). It seemed logical to break Component up so that it would be absolutely clear to the scenario designer when they created a catalog item and when they created a system for a virtual user.

Additional study and analysis of the SFT revealed that it was possible to create new reusable descriptors (henceforth called *secondary-descriptors*) from the data fields contained in existing descriptors (henceforth called *primary-descriptors*). This would result in reducing the complexity of the *primary-descriptors* that provided the data fields and would provide additional flexibility to the scenario designer through the addition of new *secondary-descriptors*. Data fields related to procedural security were taken from *Zone* and *Physical Component* to create a new *secondary-descriptor* called, *Procedural Settings*. The fact that *Procedural Settings* was an intersection of identical data fields in two *primary-descriptors* was coincidental. The decision to make *Procedural Settings* a *secondary-descriptor* was based on the amount of information it captured and the potential to save scenario designers time when developing a new scenario. Data fields related to network connections and network traffic filtering were taken from Physical Component to create two, *secondary-descriptors* named, *Network Connection* and *Filter*. All together, seventeen reusable descriptors were identified. The *primary-descriptors* and *secondary-descriptors* are listed below:

- Asset
- AssetGoal
- CatalogComponent
- Condition
- DACGroup
- Department
- Filter

- Integrity
- Network
- NetworkConnection
- PhysicalComponent
- ProceduralSettings
- Secrecy
- Trigger
- User
- Workspace
- Zone

Not all off the *primary-descriptors* were designated as reusable. Five *primary-descriptors* were so specific to a scenario that they needed to be considered each time a new scenario was created. The five non-reusable descriptors are listed below:

- Organization
- Site
- Briefing
- DebriefWin
- DebriefLose

a. The Secondary-Descriptors

A brief description of each of the newly created *secondary-descriptors* follows.

(1) Filter. The Filter descriptor attempts to capture the behavior of gateway-type devices (e.g. routers and firewalls). Filters are applied to networks, block network traffic for certain applications (e.g. ftp, http, sendmail, etc.) inbound, outbound or in both directions. Filters are applied to specific Physical Components. There can be multiple Filters for a Physical Component and therefore, multiple Filters in a scenario.

(2) Procedural Settings. The Procedural Settings descriptor is used to establish default security settings for a Physical Component and Zones. The Procedural Settings of a Zone override the Procedural Settings of a Physical Component.

Procedural Settings define minimum and maximum secrecy and integrity labels, ACLs, password policies, email, outside software, modem use, web mail, anti virus and patch management polices. There is a single Procedural Setting per Zone and Physical Component but there may be multiple Procedural Settings per scenario.

(3) Network Connection. The Network Connection descriptor is used by Physical Component to define what network the device is connected to. Network Connection requires that either ACLs or Unix like (User, Group, World) permission bits be set (the use of ACLs or permission bits is determined by the OS selected for the Physical Component) for the connection. It also requires that if a MAC-capable OS is used, then the connection be designated single level or multilevel. There can be multiple Network Connections for a Physical Component and therefore, multiple Network Connections in a scenario.

2. Human Interface

After determining which descriptors would be reusable, attention moved to design of the human interface. This phase of design was the most critical because it was the interface that would determine if scenario development became less or more complex.

The next significant step was to identify the visual elements needed to assist users when designing scenarios and how to present those elements to them. Four key components were identified that would satisfy the defined requirements: a tabbed work area, an area for a tree to represent the *reusable sets*, another tree area to represent a scenario, and a tool bar to hold components necessary to manage *reusable sets.* A fifth minor component was added to provide feedback about the tools status to the users; this feedback area is similar to the feedback area in a compiler that reports compilation errors. In addition to these application-specific interface elements, a standard menu bar and tool bar not unlike those found in most Windows applications was specified. Figure 1 below depicts the design concept for the Scenario Definition Tool.

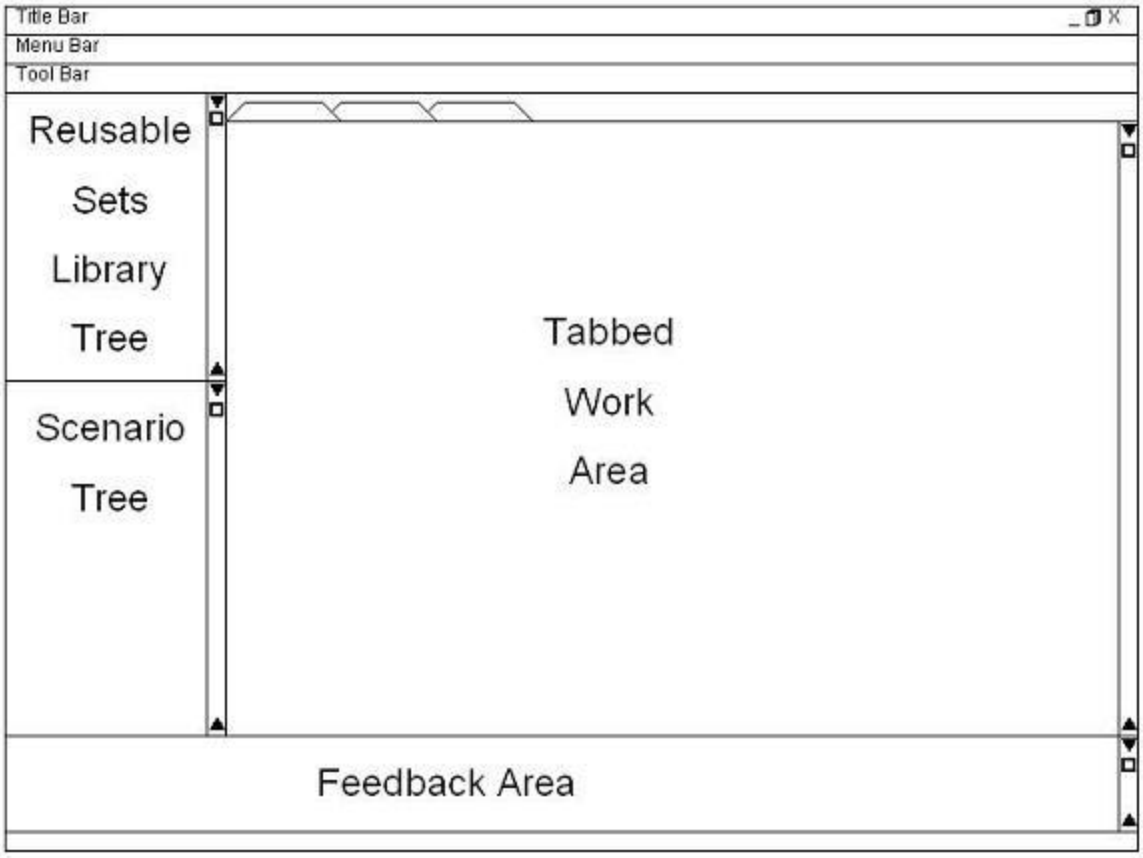

Figure 1. Design Concept for the Scenario Definition Tool

a. Reusable Sets Tree

Users need a way to see what *reusable sets* already exist. A tree is a familiar and logical way to represent the *reusable sets* library and is in keeping with the Windows paradigm for viewing lists of files in the file system. *Reusable sets* are saved to the host computer's file system, so the *reusable sets* library tree reflects the available application-specific files. The tree structure also provides the foundation for familiar Windows application features like drag and drop, double click to open and menus offering special functions that open when the right mouse button is pressed.

b. Scenario Tree

Scenario designers need a way to visualize the state of the scenario in development. A tree was again employed; this portion of the interface was called the scenario tree. The tree used in this portion of the interface did not refer to the file system

to provide information to the user but instead relied upon a data structure that was internal to the scenario. The details of this data structure will be discussed later in this chapter.

c. Tabbed Work Area

Since descriptors were going to be presented as forms, a large portion of the screen was set aside to display those forms. Since descriptors are elements of reusable *sets* it was determined that the most user-friendly way to represent these sets was to use a tab for each one.

When a user opens a reusable set, the first form in the set is presented. With each new *reusable set* opened, a new tab is opened. Anytime the SDT is run, there is always a scenario tab present. The scenario tab is always the first tab in the series and cannot be closed. In addition, there can only ever be one scenario tab open. The design restrictions on closing the scenario tab and only having one scenario tab at a time were imposed to avoid confusion about what scenario was being developed.

d. Feedback Area

In addition to the use of message dialog boxes, the feedback area provides the means to communicate the status of the SDT to the user. The feedback area reports the status of open, save and build operations.

C. SCENARIO DATA MANAGEMENT

This section of the chapter will provide information on three areas of the SDT design. First, the decision process behind the conversion of descriptors to forms will be presented. Second, the relationship between the human interface components and the data structures they represent will be presented. Finally, the relationship between the data structures of the SDT to the process of creating a Scenario Definition File will be discussed.

1. Descriptors as Forms

Part of the process of converting descriptors to forms involved deciding how the descriptors would be represented in code. Initial designs called for the form to be one object and the descriptor to be a second object. This design approach is presented in most software methodology courses as a best practice and is a valid approach to GUI design. Further analysis and design revealed that, for the SDT, this approach was not the

most efficient way to implement the GUI. Rather, the form was taken as the object and a single class was used. Two factors drove this decision. First, by making the form an object it would be possible to use native Java IO methods for saving an entire object and its state in one operation (the SDT approach to saving is discussed later in this chapter). Second, if the class/form separation approach was used it was determined that it would increase the amount of work required to maintain the SDT. This is because, there is no change that can be made to a descriptor that would not require the corresponding form (and its underlying class if the separation approach was used) to not be updated. If a field is deleted from a descriptor in a later version of the SFT then in the class/form separation approach the GUI field would have to be deleted as would the corresponding entry in the underlying class. If the class is taken as a single object (no class/form separation) then when the GUI field is deleted the update is complete. The same holds true for a field added to a descriptor.

The conversion of descriptors to forms involved some design decisions that would assure that users of the SDT would input data of a similar type in a consistent way regardless of which form they were using. Standard form elements (text fields, combo boxes, buttons etc.) were used in designing the forms.

a. Single Data Point

If the user was required to enter a single value then a standard text box was used. This data point could be a name or an integer. Figure 2 below shows a typical example.

Figure 2. Typical Single Entry

b. Single Choice

If the user was required to enter a single choice from a list of options, a standard combo box was used. There were two types of single choices, single choices from static data and single choices from dynamic data. Static data refers to data that is contained in .ini files. This information never or very rarely changes. Dynamic data refers to data based on reusable sets that have been added to scenarios in development.

Dynamic data may change with every use of the SDT and in some cases may not be present at all. Examples of single choices from static data include gender, access modes, password length etc. Figure 3 below shows a typical example.

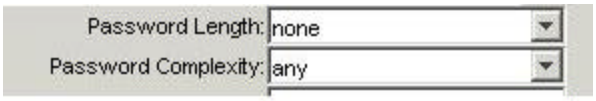

Figure 3. Static Choice

Examples of single choices from dynamic data include zone, assigned user, asset etc. Figure 4 below shows a typical example.

Figure 4. Dynamic Choice

c. Multiple Selections

If the user had the option to choose multiple entries from multiple options (regardless of whether the data was static or dynamic) then a list-to-list structure was used. Examples include choosing software that will be on a physical component (static) or which users are authorized to use a physical component (dynamic). Figure 5 below show a typical example.

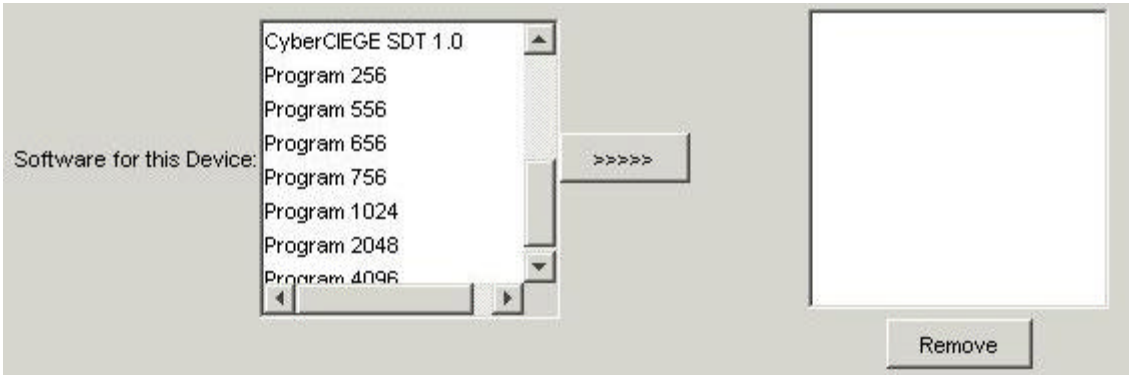

Figure 5. Multiple Selections from Multiple Options

d. Descriptions

If the user was required to provide a description, a standard text area was used as demonstrated in Figure 6.

Figure 6. Description Text Area

e. Boolean Input

If the user was required to provide a yes/no or true/false data point, a check box was used. If the user had to set one entry to true, but had to guarantee all other possibilities were false, a radio button group was used. Figure 7 depicts a typical example.

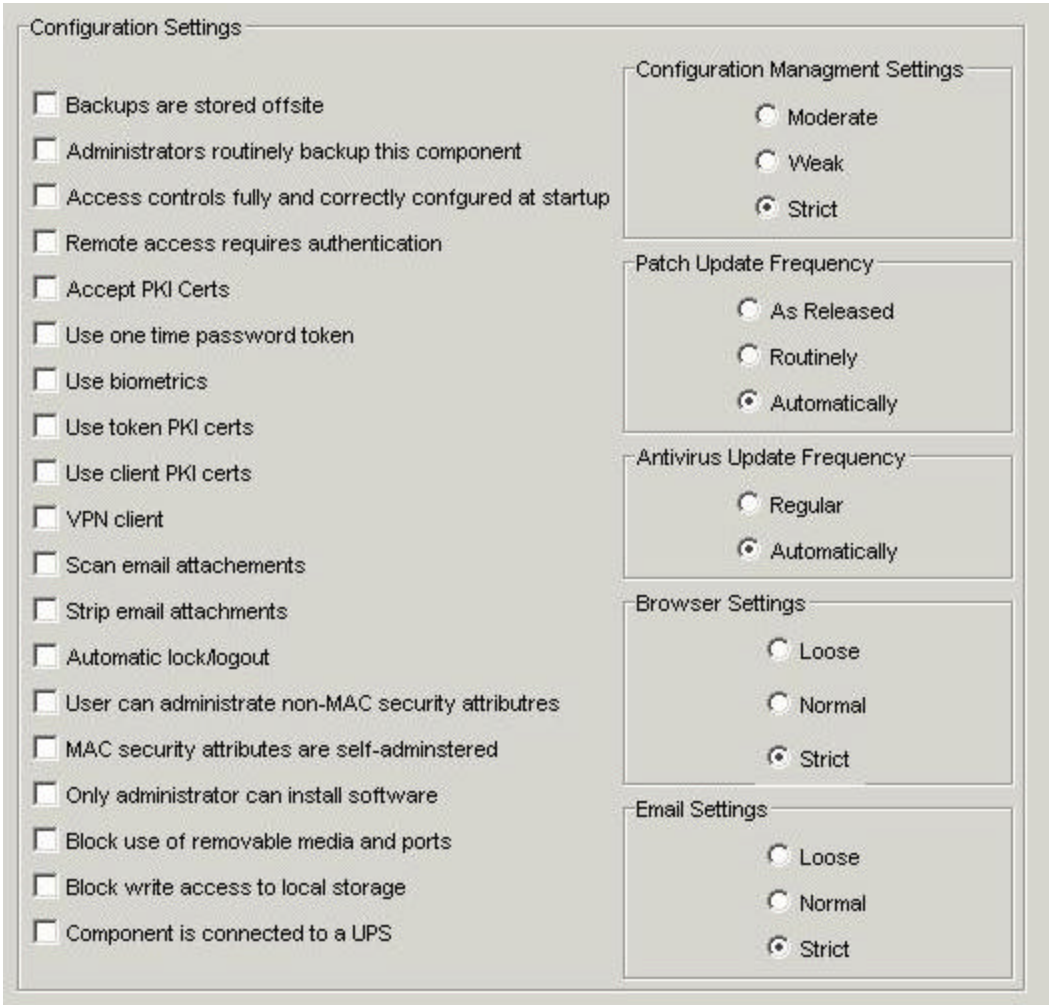

Figure 7. Boolean Data Input

2. Managing Sets

The SDT required descriptors of the same type to be collected into *reusable sets.* The requirements also specified that users must be able to navigate between elements of the set, edit those elements and delete them. These requirements are met using several components of the interface, only some of which are visible to the user. The visible elements are a combo box on the tool bar used to manage set elements, the delete button immediately to the combo box's right, the tabbed work area, and the *reusable sets* library tree. The reusable sets library tree represents the sets that have already been created and are stored in the computers file system. The tabbed work area provides the means to view each element in the set. The combo box on the tool bar allows users to manage set elements by providing features to add a new element to the set and move between

elements in the set. The delete button to the right of this combo box is used to remove the currently selected element from the set.

The content of the combo box on the tool bar is based on the currently selected tab in the tabbed work area. If the currently selected tab is the scenario tab then the combo box is empty. This is because scenarios are not stored or represented as sets. If the currently selected tab contains a reusable set then the members of that set are used to populate the combo box on the tool bar.

A *reusable set* is represented to the user as a series of forms. To the SDT a *reusable set* is an object called a *Scenario Element Set*. A *Scenario Element Set* has a data member that holds an ordered set of form objects of the same type. When a user chooses *New* from the combo box on the tool bar, a new form object is created and added to the end of the *Scenario Element Set*. To move between elements in the set, the user chooses the name of an element from the set that is listed in the combo box on the tool bar. The selection is passed to the *Scenario Element Set*, which fetches the desired form and repaints the tabbed work area with the new form. When the delete button is pressed, the same sequence of events occurs except that the desired element is deleted and the next element in the Scenario Element Set Vector is painted into the tabbed work area. The SDT does not support empty Scenario Element Sets. As a result, it is not possible to delete an element from a one-element set.

3. Managing Scenarios

Scenarios are presented to the user as a form and a tree. To the software, a scenario is a Scenario object. The form portion of the object represents all of the non-reusable descriptors in a scenario, Organization, Site, Briefing, Debrief Win and Debrief Lose.

The reusable descriptors of a scenario are stored in a data structure called the *Scenario Manager*. The *Scenario Manager* is a collection of sets of *Scenario Element Sets.* It is easier to think of the *Scenario Manager* as a filing cabinet. If the filing cabinet is the *Scenario Manger* then each drawer contains files of *Scenario Element Sets.* The index of the Scenario Manager corresponds to a particular type of reusable sets (e.g. Asset is index 0, Asset Goal is index 1, Physical Component is index 8 etc.). Returning

to the filing cabinet analogy, drawer 0 is Assets, drawer 1 is Asset Goal and drawer 8 is Physical Component. All the files in drawer 0 will be Asset Scenario Element Sets. Each page in any file of drawer 0 will be a single Asset form.

Figure 2 below depicts the structure of the Scenario Manager for *DACGroup*. Once again returning to the filing cabinet analogy, drawer 4 contains files of *DAC Group Scenario Element Sets.* The pages of the third file in the drawer (index 2) are individual *DACGroup* descriptor forms.

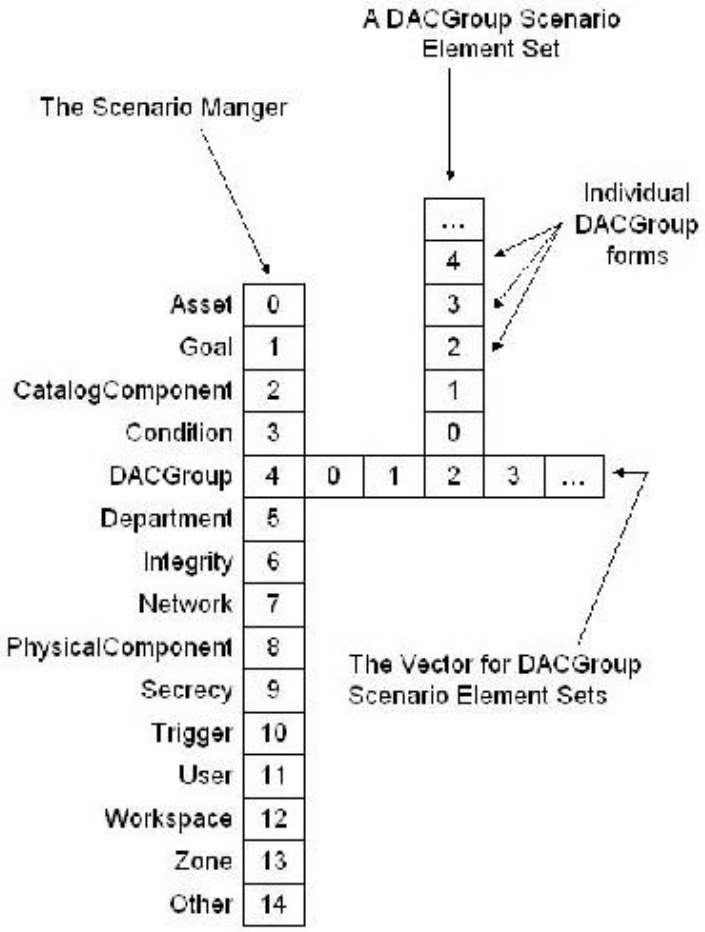

Figure 8. The Scenario Manager

The Scenario Tree reports the contents of the Scenario Manger down to the *Scenario Element Set* level. To see the individual forms within a *scenario element set*

that has been added to the scenario, the user must open it, which can be done by double clicking or right clicking the *scenario element set* file name.

To add a *reusable set* to a scenario the user can drag and drop from the *reusable sets* library, right click on a *reusable set* in the *reusable set library* and choose "add" or right click on an open tab in the tabbed work area and choose "add". The result of an add, regardless of technique, is always the same. The Scenario object determines the type of reusable set that is being added and adds it to the end of the corresponding set. In other words, no matter how one adds a *DACGroup ScenarioElementSet* it is always added to the end of the ordered set at index 4 of the Scenario Manger. *Scenario element sets* are added to the *scenario manager* in this manner for consistency.

To remove a *reusable set* from a scenario the user must right click on it and choose "remove". Based on information in the Scenario tree path the Scenario object can determine the type of the *reusable set* so it knows which Scenario Manager index to look in. Once the index of the *Scenario Manager* is determined the Scenario object opens that set and compares names until a match is found. The match is removed and the set re-indexes itself.

4. Saving State

What to save and how to save it was a key design decision. It was determined that there are two states that have to be captured in order to support reusability: the state of a *scenario element set* and the state of a scenario. The state of a *scenario element set* is captured by the files displayed in the *reusable sets library*. These files contain a single scenario element set that may contain one or more descriptor forms. The state of a scenario is captured in the form of a file called a *scenario master*. A scenario master represents a scenario and any *reusable sets* that have been added to the Scenario Manager. This dealt with the "what to save?" question but the "how to save it?" question remained.

The SDF is a formatted text file, so it seemed reasonable to explore saving scenarios and Scenario Element Sets as formatted text as well. However, this approach would make converting formatted text back to objects very complicated. The other option was to save scenarios and Scenario Element Sets as binary files. This approach

would make object regeneration very simple. Java supports writing whole objects to a file and recovering them later through its Serialization interface. Saving in this manner offered the ability to completely capture state at the moment of the save and to quickly and easily open it later for use. The advantages of this style of save outweighed the formatted text approach, so binary files became the standard for saving in the SDT.

D. SDT ORGANIZATION & CLASSES

The SDT is organized into 29 classes that can be subdivided into three categories primary, support and descriptor. The ScenarioDefinitionTool class manages the interface logic. This class provides the 22 functions that the SDT is capable of performing. All other classes serve to support this class and its functions.

1. Primary Classes

The primary classes provide the main logic and basic functions of the SDT. The following is a list of the primary classes:

- FileNode – Used by the Reusable sets Library to report the contents of the file system.
- ObjectNode – Used by the Scenario Tree to report the contents of the Scenario Manger.
- ScenarioDefinitionTool – This is the foundation for the GUI and provides all of the main logic for the program.
- ScenarioElement - The base class from which all of the descriptor classes are derived.
- ScenarioElementSet – This class is used to construct reusable sets.
- ScenarioElementSetDropTargetListener - This class makes the drop action of drag-and-drop possible.
- ScenarioElementSetTransferable – This class defines the data types that are supported for drag-and-drop operations.

2. Support Classes

The support classes provide a logical abstraction that makes the implementation of form elements to populate and manage lists easier. They make the implementation easier by providing a single object that can be stored in a Vector. The Vector can later be used to recall the object and populate the list at build time.

- AccessControlList - This class is used in any descriptor that has an Access Control List (ACL) associated with it.
- AssetGoalAsset – This class is used by AssetGoal to link Assets to access modes in the form of a list.

- CostList – This class is used by Asset to describe the cost if an access violation occurs.
- UserAssetGoal – This class is used by User to link the AssetGoal descriptor to the User descriptor and to describe virtual user's objectives for a particular Asset (via the AssetGoal).

3. Descriptor Classes

The descriptor classes are all of the forms that users populate to define a scenario. All of these classes, with the exception of Scenario, become the data members of *ScenarioElementSet*. The Scenario object is the form and logical representation of a scenario and holds all of the non-reusable descriptors. The Scenario object is also the repository for the ScenarioManager data structure is kept. A list of the descriptor classes follows:

- Asset
- AssetGoal
- CatalogComponent
- Condition
- DACGroup
- Department
- Filter
- Integrity
- Network
- NetworkConnection
- PhysicalComponent
- ProceduralSettings
- Scenario
- Secrecy
- Trigger
- User
- Workspace
- Zone

E. IMPLEMENTATION

In this section, the significant details of the implementation of the Scenario Definition Tool (SDT) will be discussed. For more detail than is provided in this chapter, the source code for the SDT is provided in Appendix B.

1. Language & IDE

The Scenario Definition Tool was developed using NetBeans 3.5.1 IDE and Java 2 Standard Edition SDK Version 1.4.2 [SUN1]. NetBeans was chosen because it is a free, open source IDE.

F. SUMMARY

This chapter has presented the design of the Scenario Definition tool. Two broad areas in the overall design were discussed: reusable sets and the human interface. The reusable sets of the SDT were identified and discussion was provided highlighting why each set was made reusable. The major components of the human interface were described and the more abstract features of the design were discussed. The major components were: the reusable sets library, the scenario tree, the tabbed work area and the feedback area. Details surrounding the approach to managing sets and scenarios as well as saving state and form layout were also provided. The next chapter will consider key implementation details and the approach to testing.

The application is organized into 29 classes. The classes can be broken down into three sets: primary classes, support classes and descriptor classes.

The SDT was implemented using NetBeans IDE 3.5.1 and Java 2 Standard edition version 1.4.2. The next chapter will describe the testing of the SDT.

IV. TESTING

A. INTRODUCTION

This chapter describes the testing process used for the Scenario Definition Tool. First the interface functions and forms that required testing are identified then the phase One (interface and forms) testing procedure is described. Then the goals of phase Two testing (functional system tests) will be discussed and that testing procedure will be described.

B. TESTING

All tests sought to achieve a single goal based on guidance provided by Fisher [Fisher 2003]. That goal was to find errors not identified during development and demonstrate that the SDT performed well. However, it is appropriate and necessary to test to ensure that the SDT is stable, forgiving of user mistakes, performs well in general, is crash resistant, and meets all its stated requirements. The complete check list used for testing is included in Appendix C.

1. Interface Functions & Forms Tests

The SDT contains 22 major interface components and 18 forms that had to be tested. The following is a list of the interface functions:

- File->New (opens a sub-menu that allows the user to choose what type of reusable set to create)
- File->Open (opens a previously saved reusable set or scenario)
- File->Save (saves the reusable set contained in the currently selected tab)
- File->Save Scenario (saves a scenario)
- File->Save All (saves all open reusable sets and the scenario)
- File->Save As (saves the reusable set contained in the currently selected tab under a new name)
- File->Save Scenario As (saves the current scenario under a new name)
- File->Exit (exits the application)
- Tools->Build (causes the SDF to be created)
- Set Element Management Combo Box->New (creates a new form in the reusable set contained in the currently selected tab)
- Set Element Management Combo Box->switch between elements in a set (allows movement between the forms of the reusable set contained in the currently selected tab)
- Set Element Management Combo Box->Delete Button (deletes the currently displayed form from the reusable set contained in the currently selected tab)

- Reusable Sets Library Tree->double click to open file (opens the reusable set that is double clicked)
- Reusable Sets Library Tree Right Click Menu->Open (opens the reusable set that is selected)
- Reusable Sets Library Tree Right Click Menu->Add (adds the selected reusable set to the current scenario)
- Reusable Sets Library Tree Right Click Menu->Delete (deletes the selected reusable set from the file-system)
- Scenario Tree->double click to open file (opens the reusable set that is double clicked)
- Scenario Tree Right Click Menu->Open (opens the reusable set that is selected)
- Scenario Tree Right Click Menu->Remove (removes the reusable set that is selected from the current scenario)
- Tabbed Work Area Right Click Menu->Add (adds the reusable set contained in the currently selected tab to the current scenario)
- Tabbed Work Area Right Click Menu->Remove (closes the currently selected tab)
- Drag-and-Drop (adds the selected reusable set to the current scenario)

The following is a list of the forms in the SDT:

- Scenario
- Workspace
- CatalogComponent
- Network
- Department
- DACGroup
- Secrecy
- Integrity
- Asset
- AssetGoal
- User
- NetworkConnection
- ProceduralSettings
- Zone
- PhysicalComponent
- Filter
- Condition
- Trigger

To make the testing thorough, it was determined that some amount of stress would have to be put on the interface and forms. It seemed reasonable that this stress should cause some previously unidentified errors to surface and would simulate a reasonable workload. As a result, three sets of testing parameters were defined. The first

set had all data fields empty, the second set had some random fields empty and the third had no fields empty. Finally, to the extent that it was possible the tests would have to cover all functions for all forms. Scenario and Save All were identified as two problem areas for testing.

a. Test Procedure

A standard test procedure was developed and used on all forms. The procedure required that, for each form, each interface function should be performed on that form. Each test had a pass, fail or not applicable (n/a) outcome. The "not applicable" outcome was added because many of the interface functions have no effect on Scenario. Similarly, Scenario-specific interface functions have no effect on the *reusable sets.* By having an "n/a" outcome for each test, the tests could remain standard without Scenario failing several tests or having to create special tests just for Scenario and vice versa for the *reusable sets* with regard to the Scenario-specific interface functions.

In addition to the interface function that was the focus of each test, other form-specific tests were performed depending on which interface function was being tested. For example, if a "save" operation was performed the tester would be asked to check the "Reusable Sets Library Tree" and the "operating systems file-system" to make sure that the "save" really worked. Any time a set or form was repainted (by an "open", "save" or "switch") to make testing complete the form's interface elements were checked to make sure they were still functioning.

All tests had a catchall test, "No other errors occurred" and a comments field. This provided the opportunity to note seemingly peculiar behavior. For example, if a null pointer exception occurred that was not induced/stimulated by a specific test the error still had to be noted. The test procedure combined with comments regarding the error encountered provided enough information to the developer to recreate the error and fix it.

Testing of scenario definition file generation was not performed in this set of tests. This test was reserved for the functional tests of the completed SDT.

2. Functional Tests

The goal of the functional tests was to determine if the tool could generate a properly formatted Scenario Definition File. This test determined that the application met its primary requirement.

a. Test Procedure

To prepare for the test, two hand-written SDFs were created. Hand written SDFs were used simply because there was no viable software alternative to generate an SDF that was known to be correct and meet the SFT Version 14 standard.

The passing criterion for the test was that the SDT generated an SDF that matched the hand written standard with a minimal number of differences evident through both visual inspection and based on the results of running a difference generator on the files. Some differences were simply unacceptable and would be categorized as a failure of the tool, while other differences were deemed acceptable. Criteria for acceptable and unacceptable differences are given below.

Unacceptable differences:

- The omission of any descriptor or field that appeared in the crafted standard
- Major formatting errors (e.g. omission of a new line where it was expected)
- A comment without the leading '//'
- A generated SDF grossly different from the standard

Acceptable differences:

- All the text in a long description is not perfectly indented
- Minor formatting errors (e.g. indention does not perfectly match)

All differences were noted for correction.

C. SUMMARY

Testing was conducted in two phases. Phase One involved testing the 22 major interface functions and the 18 forms used by the interface. A standard test was used that applied each interface function to each form. Phase two testing involved testing the SDTs ability to create an SDF that met SFT Version 14 standards. The next chapter will discuss future work and conclude this thesis.

V. FUTURE WORK & CONCLUSION

A. FUTURE WORK

In any ambitious project, there are always more good ideas than there are resources to implement them, and the Scenario Definition Tool project is no exception. Once development started, many excellent ideas emerged. As many of those ideas as possible were integrated into the SDT, while the others were set aside as future work largely due to time constraints. One of the reasons time became an issue was that this was the author's first in depth experience with the Java programming language. Precious time was required to learn how to implement some of the more complicated interface components (such as trees and drag-and-drop). In the end, Version 1.0 of the Scenario Definition Tool was successfully implemented. The rest of this section is dedicated to future improvements that could be made to the SDT.

1. Interface Upgrades

Several interface upgrades were proposed during development. The first of those upgrades was to convert the interface from the static separator interface implemented now to a dynamic multiple document interface (MDI) similar to those found in compilers and high-end applications.

The current implementation of the SDT prompts the user for a scenario name and designer name at startup, then opens the application with a new, empty scenario bearing the name just entered by the user. A future implementation should provide a file-system browser at startup that allows the user to browse for a particular scenario to be loaded at startup.

A simple proposed upgrade would maintain state by recording the last element/form used in a reusable set and open to that element/form when the set was later re-opened. Another simple upgrade along the same lines would have a scenario open all the sets added to it when the scenario is opened.

Scenario Designers should be able to create a scenario master directory. The purpose of this directory would be to allow a designer to put all scenario masters for a related series of scenarios in one place. Expanding on this idea further, there should be a

"Build All" function added to the Tools menu in addition to Build. Build All would allow the designer to specify a scenario master directory and build all of the scenarios in that directory in one call rather than by making multiple calls.

In the current implementation, all SDFs go to a predetermined directory after a build. A future improvement would allow an SDF to be opened in a native text editor (e.g. WordPad, Notepad, VI, Emacs, etc.) by selecting View SDF from the Open menu. Another proposed upgrade would allow the user to determine the final destination of the SDF after it is generated.

Another proposed upgrade would enable the Workspace form to write the workspace text file used by the game engine.

As currently implemented, the SDT does not format the contents of Access Control List or Cost List, list boxes, and the SDT also does not allow for the editing of entries in the previously mentioned list boxes. Two proposed upgrades would, one, format the contents of the list boxes or convert them to tables, and two, make it possible to click on an entry in the list box and edit the selection.

2. Upgrade to the Latest SFT Standard

Work continued on scenario language and the Scenario Format Template while the SDT was being developed. As a result, the SDT is not current with the latest versions of the SFT and needs to be updated.

Part of this update would include making Filter a *primary-descriptor* so that it is consistent with the new SFT.

3. More Robust Error Handling

Version 1.0 of the Scenario Definition Tool has limited error handling capabilities and no range testing. The SDT has enough error handling to make it stable and usable. However, users can literally put any value into a text field and the application will accept it. The goal of the thesis was not to create a perfect scenario generator, but to demonstrate that the syntax complexity of the SFT could be managed, that scenario generation could be automated and to solve a real world problem. Therefore, a key improvement that is needed is implementation of robust error handling and range testing. Version 14 of the SFT specifies what values are expected and what ranges are allowed.

After the SDT's error handling and range testing are implemented, a new battery of tests will be necessary to demonstrate that the limits are enforced.

4. Implement the Student Assessment XML Tags

The thesis work of another student, Tiat Lang Teo [Teo 2002] involved the addition of XML tags to the SDF to support a student assessment module for instructors. The current implementation of the SDT does not add these tags. The tool should be extended to support student assessment.

5. Help Menu

The current SDT does not implement a "help" menu item. While this thesis will provide a basic user manual, the addition of help features to the tool would be a welcome addition.

6. Split User

User should be split into two descriptors in the same way that Component was split. User should become User and Support staff.

B. CONCLUSION

This thesis set out to demonstrate that it is possible to manage the complexity of CyberCIEGE Scenario Definition language and that advances could be made toward automating scenario generation if scenario descriptors were reusable. It succeeded on both counts. Software now exists that abstracts the complexity of scenario language enough that scenario designers need only limited or no knowledge of the scenario language to build a scenario. Moreover, with reusable descriptor sets, a scenario designer can easily build a series of related scenarios based on one or more common *reusable sets.* If a change is made to one of the common *reusable sets*, the designer can regenerate the scenarios and the change will automatically implemented in all of them. All of this can be done without ever opening a text editor and without having to worry about the accuracy of the resulting SDFs.

As scenario generation becomes more automated, CyberCIEGE is likely to see greater use. This broadening of CyberCIEGE usage will expose greater numbers of students, corporate employees and game players to Information Assurance and will help

the CyberCIEGE project achieve its educational goals. It is the author's sincere desire that the work of automating CyberCIEGE scenario generation will continue where this work has ended.

APPENDIX A: USER MANUAL

A. INTRODUCTION

This document will provide basic guidance for using Version 1.0 of the Scenario Definition Tool (SDT).

B. STARTING THE SCENARIO DEFINITION TOOL

Since the SDT is still in development, its operation is closely tied to the NetBeans IDE. Future updates will include the ability to use the SDT without the support of the NetBeans IDE. The following instructions assume that NetBeans is running and that the SDT project is loaded.

If the SDT needs to be compiled make sure the ScenarioDefinitionTool.java file is loaded in the editing tab, click anywhere in the source code and press F11. When the compiler is finished, click in the editor window again and press F6 (Note: if the SDT does not need to be compiled then this is the only step needed to launch the application).

A successful startup of the SDT is indicated by a prompt that asks for a scenario file name. The name provided will be used to name a brand new scenario that is created during startup. Once the file name is provided another prompt will open asking for the scenario designer's name. After the scenario designer's name is provided the rest of the application will open.

1. The SDT Interface

Figure 9 below shows the SDT interface. Besides the file bar and tool bar, the SDT interface is broken up into 4 areas: the reusable sets library tree, the scenario tree, the tabbed work area and the feedback area.

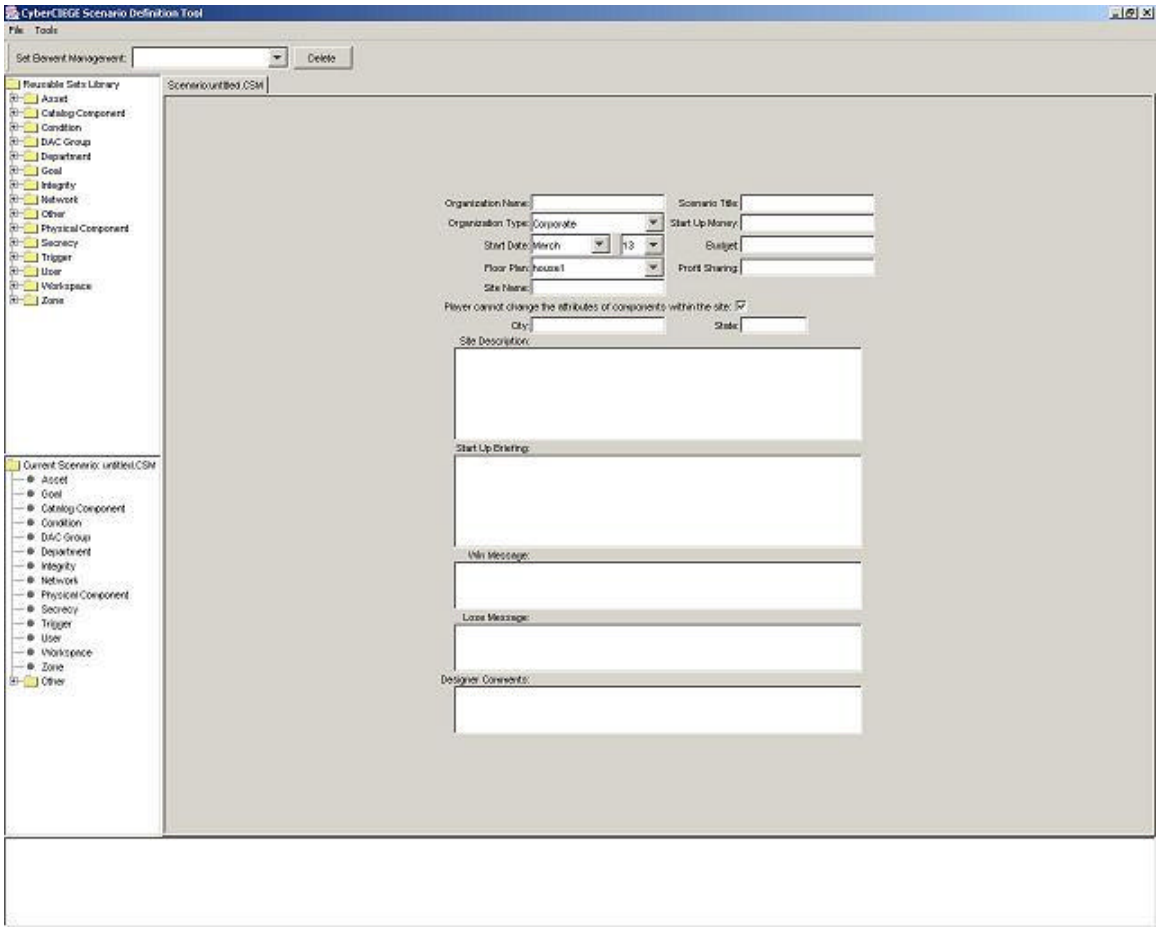

Figure 9. The CyberCIEGE Scenario Definition Tool

a. The Menu Bar

A standard menu bar is located along the top of the interface. There are two entries on the menu bar: File and Build. The basic purpose of the File menu is to provide the functionality to create new reusable sets, save reusable sets and scenarios and to exit the application. The basic purpose of the Build menu is to provide the functionality to generate a Scenario Definition File (SDF). The specific functions of these menus are discussed throughout this document.

b. The Tool Bar

Located directly below the menu bar is a tool bar that contains two elements: a combo box and a button. The combo box is used to create new set elements and to move between the members of a reusable set. The Delete button to the combo box's right is used to delete elements from a reusable set.

c. The Reusable Sets Library

The reusable sets library is located in the upper left hand corner of the interface just below the tool bar (see Figure 9 above). The reusable sets library represents all of the reusable sets that have been saved and that can be added to a scenario. It is organized by descriptor.

d. The Scenario Tree

The scenario tree is located directly below the reusable sets library and represents the current scenario in development. It shows any reusable sets that have been added to the current scenario.

e. The Tabbed Work Area

The tabbed work area is located directly to the right of the reusable sets library tree and the scenario tree. This is where the forms contained in reusable sets are individually displayed.

f. The Feedback Area

The feedback area is located along the bottom of the interface. The feedback area provides information to the scenario designer regarding the status and operation of the SDT.

C. REUSABLE SET MANAGEMENT

The SDT implements an approach to scenario development based on sets of scenario descriptor forms that can be used repeatedly. These reusable sets are a collection of descriptor forms of the same type (e.g. all DAC Group forms).

1. Creating New Reusable Sets

To create a new reusable set go to the “File” menu and select “New”. A sub-menu will open that lists all of the descriptor types. Select the desired descriptor type to create a reusable set for that descriptor. A prompt will open asking for the name of the new set. The name provided will be used to name the file associated with the new reusable set. Once a file name is provided, a new tab will open in the tabbed work area with a blank descriptor form ready for input. This form is the first element in the reusable set

2. Creating New Reusable Set Elements

To create a new element in an existing reusable set, first, select the tab that contains the reusable set of interest. Second, Select "New" from the combo box on the tool bar. A new, blank, descriptor form will be added to the reusable set in the active tab.

3. Moving Between Reusable Set Elements

To move between elements in a reusable set use the combo box on the tool bar. First, select the tab that contains the reusable set of interest. Second, click on the combo box. If the combo box is empty (except for the "New" item) then one of two things has happened. One, the combo box needs to be updated or two, the set only has one as of yet un-named entry. To update the combo box select any other tab and then re-select the tab of interest. If the reusable set has any elements they will now be displayed in the combo box. To move to the new member simply choose it from the list. The tabbed work area will update with the contents of the chosen element.

4. Deleting Reusable Set Elements

To delete elements in a reusable set, start with the combo box on the tool bar. First, select the tab that contains the reusable set of interest. Second, click on the combo box. If the combo box is empty (except for the "New" item) then one of two things has happened. One, the combo box needs to be updated or two, the set only has one as of yet un-named entry. To update the combo box select any other tab and then re-select the tab of interest. If the reusable set has any elements, they will now be displayed in the combo box. Choose the name of the element to be deleted from the list. Click the delete button to the right of the combo box. A dialog box will open requiring confirmation of the deletion. If "yes" is selected the element will be deleted from the set. The tabbed work area will update with previous element in the set displayed. NOTE: The first element in any set cannot be deleted only edited.

5. Saving Reusable Sets

To save a reusable set, first, select the tab that contains the reusable set of interest. Then go to the "File" menu and select "Save" or "Save As". If "Save" is chosen the feedback area will report the progress of the save. If "Save As" is chosen, a prompt will open asking for the new file name to be used. Once the new file name is provided, the feedback area will report the progress of the "save as" operation.

6. Opening Existing Reusable Sets

There are five ways to open an existing reusable set.

a. File->Open

Go to the "File" menu and select "Open". A file chooser dialog box will open with a list of the descriptor types. Double click on the descriptor type that corresponds to the reusable set to be opened. The file chooser dialog box will update with a list of all of the files for that descriptor. Double click on the desired file. The tabbed work area will update with a new tab and the first element of the newly opened reusable set will be presented.

b. Reusable Library Tree Double Click

Expand the reusable library tree for the descriptor type that corresponds to the reusable set to be opened. The tree will update with a list of all of the files for that descriptor. Double click on the desired file. The tabbed work area will update with a new tab and the first element of the newly opened reusable set will be presented.

c. Reusable Library Tree Right Mouse Button Menu

Expand the reusable library tree for the descriptor type that corresponds to the reusable set to be opened. The tree will update with a list of all of the files for that descriptor. Right click on the desired file. A popup menu will launch, choose the "Open" option from the menu. The tabbed work area will update with a new tab and the first element of the newly opened reusable set will be presented.

d. Scenario Tree Double Click

Expand the scenario tree for the descriptor type that corresponds to the reusable set to be opened. The tree will update with a list of all of the files for that descriptor. Double click on the desired file. The tabbed work area will update with a new tab and the first element of the newly opened reusable set will be presented.

e. Scenario Tree Right Mouse Button Menu

Expand the scenario tree for the descriptor type that corresponds to the reusable set to be opened. The tree will update with a list of all of the files for that descriptor. Right click on the desired file. A popup menu will launch, choose the "Open" option from the menu. The tabbed work area will update with a new tab and the first element of the newly opened reusable set will be presented.

7. Closing Reusable Sets

To close a reusable set, right click on the tab that contains the reusable set to be closed. A popup menu will launch, choose the "Remove" option from the menu and the tab will be closed. NOTE: Any unsaved changes to that reusable set will be lost.

8. Deleting Reusable Sets

To permanently remove a reusable set from the application and the underlying system, expand the reusable library tree for the descriptor type that corresponds to the reusable set to be deleted. The tree will update with a list of all of the files for that descriptor. Right click on the desired file. A popup menu will launch, choose the "Delete" option from the menu. A dialog box will open confirming that the deletion has occurred.

9. Dealing with Descriptor Dependencies

Several of the descriptors are dependent on other descriptors for their content. If the process of completing a descriptor form is not approached in such a way as to satisfy dependencies, then the process can be very frustrating. This section will offer a suggested procedure to creating and adding reusable sets to a scenario so that all dependencies can be satisfied. For instructions on how to add reusable sets to a scenario see section D.2 below.

Descriptors that have dependencies reference the scenario in development to satisfy those dependencies. If reusable sets are added out of sequence, the lists or combo boxes that represent dependent information will be blank. If a dependency has been met but a list or combo box is still empty close the reusable set and reopen it.

The following procedure can be applied to reusable sets as they are created and/or added to a scenario:

1. Workspace
2. Catalog Component
3. Network
4. Department
5. DAC Group
6. Secrecy
7. Integrity
8. Asset (do not complete the Access Control List (ACL) entry at this time – Asset has been added at this time to satisfy a dependency in Asset Goal)
9. Asset Goal

10. Remove all Assets from the scenario (they will be added again later)
11. User
12. Add the Assets again (the user dependency of the ACL has now been met – Asset was added before to satisfy a dependency in Asset Goal)
13. Network Connection
14. Procedural Settings
15. Zone
16. Physical Component
17. Condition
18. Trigger

D. SCENARIO MANAGEMENT

1. Creating New Scenarios

A new scenario can only be created when the SDT is started. If a new scenario is desired while the application is in use, save any unsaved reusable sets and restart the application. Scenario creation is managed in this way to avoid confusion about which scenario is being developed at a given time. The SDT allows only one scenario to be open at a time.

2. Adding Reusable Sets to a Scenario

There are three ways to add a reusable set to a scenario.

a. Reusable Library Tree Right Mouse Button Menu

Expand the reusable library tree for the descriptor type that corresponds to the reusable set to be added. The tree will update with a list of all of the files for that descriptor. Right click on the desired file. A popup menu will launch, choose the "Add" option from the menu. The scenario tree will update to reflect the newly added reusable set.

b. Tabbed Work Area Tab Right Mouse Button Menu

Right click the tab that contains the reusable set to be added. A popup menu will launch, choose the "Add" option from the menu. The scenario tree will update to reflect the newly added reusable set. NOTE: when a reusable set is added in this manner the reusable set is automatically saved. This is not necessary for the other two add techniques as those reusable sets are already saved.

c. Drag and Drop

Expand the reusable library tree for the descriptor type that corresponds to the reusable set to be added. Click on the file to be added and hold the mouse button down. Now drag the file to the scenario tree and release the mouse button. The scenario tree will update to reflect the newly added reusable set.

3. Saving Scenarios

To save a scenario, first, select the scenario tab. Then go to the "File" menu and select "Save Scenario" or "Save Scenario As". If "Save Scenario" is chosen the feedback area will report the progress of the save. If "Save As" is chosen, a prompt will open asking for the new file name to be used. Once the new file name is provided, the feedback area will report the progress of the save as operation.

4. Opening Existing Scenarios

Go to the "File" menu and select "Open". A file chooser dialog box will open with a list of the descriptor types. Navigate up one directory so that the files listed include, Reusable Sets Library, Scenarios and SDF. Double click on the Scenarios file. The file chooser dialog box will update with a list of all of the scenarios. Double click on the desired file. The tabbed work area will update the scenario tab to reflect the newly opened scenario. The scenario tree will also update to reflect the newly opened scenario.

5. Removing Reusable Sets from a Scenario

To remove a reusable set from the scenario in development, expand the scenario tree for the descriptor type that corresponds to the reusable set to be removed. The tree will update with a list of all of the files for that descriptor. Right click on the desired file. A popup menu will launch, choose the "Remove" option from the menu. Once "Remove" is selected the scenario tree will update to reflect the change. To make the removal permanent, it is necessary to save the scenario.

E. GENERATING SCENARIO DEFINITION FILES

To create a scenario definition file (SDF) at any time during a scenario's development go to "Build" on the menu bar and choose the "Build" menu item. To ensure the most accurate and recent information goes into the build, choose "File" from the file menu and then "Save All".

APPENDIX B: SOURCE CODE

A. THE SOURCE OF: ACCESS CONTROL LIST

```
/*
 * AccessControlList.java
 *
 * Created on February 2, 2004, 2:03 PM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is used as an intermediate container to store user choices for
 *an ACL.
 */
public class AccessControlList implements java.io.Serializable
{
    public AccessControlList()
    {
        user = new String();
        group = new String();
        read = new String();
        write = new String();
        control = new String();
        execute = new String();
    }

    public void setUser(String aUser)
    {
        user = aUser;
    }
    public void setGroup(String aGroup)
    {
        group = aGroup;
    }
    public void setRead(String aRead)
    {
        read = aRead;
    }
    public void setWrite(String aWrite)
    {
        write = aWrite;
    }
    public void setControl(String aControl)
    {
        control = aControl;
    }
    public void setExecute(String aExecute)
    {
        execute = aExecute;
    }
    public String getUser()
    {
        return user;
    }
    public String getGroup()
    {
```

```
        return group;
    }
    public String getRead()
    {
        return read;
    }
    public String getWrite()
    {
        return write;
    }
    public String getControl()
    {
        return control;
    }
    public String getExecute()
    {
        return execute;
    }
    public String getACL()
    {
        return
"(USER.GROUP)"+user+"."+group+"(READ)"+read+"(WRITE)"+write+"(CONTROL)"+control+"(EXECUTE)"+ex
ecute;
    }

    private String user;
    private String group;
    private String read;
    private String write;
    private String control;
    private String execute;
}
```

B. THE SOURCE OF: ASSET

```
/*
 * Asset.java
 *
 * Created on January 22, 2004, 3:24 PM
 */

package CCSDT;
import java.util.*;
import javax.swing.*;
import java.io.*;
/**
 *
 * @author  kjohns
 *This class is the graphical representation of an Asset descriptor.  This
 *is where the Asset form is managed and the Asset toString() resides.
 */
public class Asset extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Asset */
    public Asset()
    {
        initComponents();
        //acl_added is used by toString to determine whether or not to print
        //the acl
        acl_added = false;
        //these two vectors are used to store the users acl and cost list
```

```
    //selections
    aclTotalAccessListVec = new Vector();
    costListTotalListVec = new Vector();
    //the following combo box entries initialize the combo boxes
    aclControlComboBox.addItem("Y");
    aclControlComboBox.addItem("N");
    aclControlComboBox.addItem("X");
    aclExecuteComboBox.addItem("Y");
    aclExecuteComboBox.addItem("N");
    aclExecuteComboBox.addItem("X");
    aclReadComboBox.addItem("Y");
    aclReadComboBox.addItem("N");
    aclReadComboBox.addItem("X");
    aclWriteComboBox.addItem("Y");
    aclWriteComboBox.addItem("N");
    aclWriteComboBox.addItem("X");

    costReadComboBox.addItem("-");
    costReadComboBox.addItem("X");
    costWriteComboBox.addItem("-");
    costWriteComboBox.addItem("X");
    costControlComboBox.addItem("-");
    costControlComboBox.addItem("X");
    costExecuteComboBox.addItem("-");
    costExecuteComboBox.addItem("X");

    assetNameComboBox.addItem("$user");
    assetNameComboBox.addItem("$default_group");
    assetNameComboBox.addItem("$secrecy_clearance");
    assetNameComboBox.addItem("$integrity_clearance");
}

private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;

    assetScrollPane = new javax.swing.JScrollPane();
    assetPanel = new javax.swing.JPanel();
    aclPanel = new javax.swing.JPanel();
    aclUserLabel = new javax.swing.JLabel();
    aclGroupLabel = new javax.swing.JLabel();
    aclReadLabel = new javax.swing.JLabel();
    aclWriteLabel = new javax.swing.JLabel();
    aclControlLabel = new javax.swing.JLabel();
    aclExecuteLabel = new javax.swing.JLabel();
    aclUserComboBox = new javax.swing.JComboBox();
    aclUserFinalTextField = new javax.swing.JTextField();
    aclGroupComboBox = new javax.swing.JComboBox();
    aclGroupFinalTextField = new javax.swing.JTextField();
    aclReadComboBox = new javax.swing.JComboBox();
    aclWriteComboBox = new javax.swing.JComboBox();
    aclControlComboBox = new javax.swing.JComboBox();
    aclExecuteComboBox = new javax.swing.JComboBox();
    aclAddButton = new javax.swing.JButton();
    aclScrollPane = new javax.swing.JScrollPane();
    aclList = new javax.swing.JList();
    aclRemoveButton = new javax.swing.JButton();
    boolPanel = new javax.swing.JPanel();
    isInstaniatedCheckBox = new javax.swing.JCheckBox();
    hasDACCheckBox = new javax.swing.JCheckBox();
    textAndComboPanel = new javax.swing.JPanel();
    nameLabel = new javax.swing.JLabel();
    nameTextField = new javax.swing.JTextField();
```

```
secrecyLabel = new javax.swing.JLabel();
secrecyComboBox = new javax.swing.JComboBox();
secrecyFinalTextField = new javax.swing.JTextField();
integrityLabel = new javax.swing.JLabel();
integrityComboBox = new javax.swing.JComboBox();
integrityFinalTextField = new javax.swing.JTextField();
dosMotiveLabel = new javax.swing.JLabel();
dosMotiveTextField = new javax.swing.JTextField();
availabilityPenaltyLabel = new javax.swing.JLabel();
availabilityPenaltyTextField = new javax.swing.JTextField();
assetNameComboBox = new javax.swing.JComboBox();
costListPanel = new javax.swing.JPanel();
costUserLabel = new javax.swing.JLabel();
costGroupLabel = new javax.swing.JLabel();
costReadLabel = new javax.swing.JLabel();
costWriteLabel = new javax.swing.JLabel();
costControlLabel = new javax.swing.JLabel();
costExecuteLabel = new javax.swing.JLabel();
costUserComboBox = new javax.swing.JComboBox();
costUserFinalTextField = new javax.swing.JTextField();
costGroupComboBox = new javax.swing.JComboBox();
costGroupFinalTextField = new javax.swing.JTextField();
costReadComboBox = new javax.swing.JComboBox();
costWriteComboBox = new javax.swing.JComboBox();
costControlComboBox = new javax.swing.JComboBox();
costExecuteComboBox = new javax.swing.JComboBox();
costAddButton = new javax.swing.JButton();
costScrollPane = new javax.swing.JScrollPane();
costList = new javax.swing.JList();
costRemoveButton = new javax.swing.JButton();
costValuesPanel = new javax.swing.JPanel();
costLabel = new javax.swing.JLabel();
costTextField = new javax.swing.JTextField();
costChangeTextField = new javax.swing.JTextField();
costChangeLabel = new javax.swing.JLabel();
costAttackerMotiveLabel = new javax.swing.JLabel();
costAttackerMotiveTextField = new javax.swing.JTextField();
costAttackerValueChangeTextField = new javax.swing.JTextField();
costAttackerValueChangeLabel = new javax.swing.JLabel();

setLayout(new java.awt.GridLayout(1, 0));

assetPanel.setLayout(new java.awt.GridBagLayout());

assetPanel.setPreferredSize(new java.awt.Dimension(821, 814));
aclPanel.setLayout(new java.awt.GridBagLayout());

aclPanel.setBorder(new javax.swing.border.TitledBorder("Access Control List"));
aclUserLabel.setText("User");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
aclPanel.add(aclUserLabel, gridBagConstraints);

aclGroupLabel.setText("Group");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
aclPanel.add(aclGroupLabel, gridBagConstraints);

aclReadLabel.setText("Read");
gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
aclPanel.add(aclReadLabel, gridBagConstraints);

aclWriteLabel.setText("Write");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
aclPanel.add(aclWriteLabel, gridBagConstraints);

aclControlLabel.setText("Control");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 3;
aclPanel.add(aclControlLabel, gridBagConstraints);

aclExecuteLabel.setText("Execute");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
aclPanel.add(aclExecuteLabel, gridBagConstraints);

aclUserComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclUserComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aclUserComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
aclPanel.add(aclUserComboBox, gridBagConstraints);

aclUserFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
aclPanel.add(aclUserFinalTextField, gridBagConstraints);

aclGroupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aclGroupComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
aclPanel.add(aclGroupComboBox, gridBagConstraints);

aclGroupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
aclPanel.add(aclGroupFinalTextField, gridBagConstraints);

aclReadComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
```

```
aclPanel.add(aclReadComboBox, gridBagConstraints);

aclWriteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
aclPanel.add(aclWriteComboBox, gridBagConstraints);

aclControlComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 4;
aclPanel.add(aclControlComboBox, gridBagConstraints);

aclExecuteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 4;
aclPanel.add(aclExecuteComboBox, gridBagConstraints);

aclAddButton.setText("Add");
aclAddButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aclAddButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclAddButton, gridBagConstraints);

aclScrollPane.setPreferredSize(new java.awt.Dimension(700, 150));
aclScrollPane.setViewportView(aclList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclScrollPane, gridBagConstraints);

aclRemoveButton.setText("Remove");
aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aclRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclRemoveButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(20, 0, 20, 0);
assetPanel.add(aclPanel, gridBagConstraints);
```

```
boolPanel.setLayout(new java.awt.GridBagLayout());

boolPanel.setBorder(new javax.swing.border.TitledBorder("Initially On Component"));
boolPanel.setPreferredSize(new java.awt.Dimension(172, 101));
isInstaniatedCheckBox.setText("Is Instaniated");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel.add(isInstaniatedCheckBox, gridBagConstraints);

hasDACCheckBox.setText("Has DAC");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel.add(hasDACCheckBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.insets = new java.awt.Insets(0, 50, 0, 0);
assetPanel.add(boolPanel, gridBagConstraints);

textAndComboPanel.setLayout(new java.awt.GridBagLayout());

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Asset Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(nameTextField, gridBagConstraints);

secrecyLabel.setText("Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(secrecyLabel, gridBagConstraints);

secrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
secrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        secrecyComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(secrecyComboBox, gridBagConstraints);

secrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
textAndComboPanel.add(secrecyFinalTextField, gridBagConstraints);

integrityLabel.setText("Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(integrityLabel, gridBagConstraints);

integrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
integrityComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        integrityComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(integrityComboBox, gridBagConstraints);

integrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(integrityFinalTextField, gridBagConstraints);

dosMotiveLabel.setText("Denial Of Service Motive:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(dosMotiveLabel, gridBagConstraints);

dosMotiveTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(dosMotiveTextField, gridBagConstraints);

availabilityPenaltyLabel.setText("Availablity Penalty:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(availabilityPenaltyLabel, gridBagConstraints);

availabilityPenaltyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(availabilityPenaltyTextField, gridBagConstraints);

assetNameComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
assetNameComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetNameComboBoxActionPerformed(evt);
    }
```

```
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(assetNameComboBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
assetPanel.add(textAndComboPanel, gridBagConstraints);

costListPanel.setLayout(new java.awt.GridBagLayout());

costListPanel.setBorder(new javax.swing.border.TitledBorder("Cost List"));
costListPanel.setPreferredSize(new java.awt.Dimension(810, 320));
costUserLabel.setText("User");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
costListPanel.add(costUserLabel, gridBagConstraints);

costGroupLabel.setText("Group");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
costListPanel.add(costGroupLabel, gridBagConstraints);

costReadLabel.setText("Read");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
costListPanel.add(costReadLabel, gridBagConstraints);

costWriteLabel.setText("Write");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
costListPanel.add(costWriteLabel, gridBagConstraints);

costControlLabel.setText ("Control");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 3;
costListPanel.add(costControlLabel, gridBagConstraints);

costExecuteLabel.setText("Exe cute");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
costListPanel.add(costExecuteLabel, gridBagConstraints);

costUserComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
costUserComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      costUserComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
```

```
gridBagConstraints.gridy = 4;
costListPanel.add(costUserComboBox, gridBagConstraints);

costUserFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
costListPanel.add(costUserFinalTextField, gridBagConstraints);

costGroupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
costGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        costGroupComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
costListPanel.add(costGroupComboBox, gridBagConstraints);

costGroupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
costListPanel.add(costGroupFinalTextField, gridBagConstraints);

costReadComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
costListPanel.add(costReadComboBox, gridBagConstraints);

costWriteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
costListPanel.add(costWriteComboBox, gridBagConstraints);

costControlComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 4;
costListPanel.add(costControlComboBox, gridBagConstraints);

costExecuteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 4;
costListPanel.add(costExecuteComboBox, gridBagConstraints);

costAddButton.setText("Add");
costAddButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        costAddButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
```

```
costListPanel.add(costAddButton, gridBagConstraints);

costScrollPane.setPreferredSize(new java.awt.Dimension(700, 150));
costScrollPane.setViewportView(costList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
costListPanel.add(costScrollPane, gridBagConstraints);

costRemoveButton.setText("Remove");
costRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        costRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
costListPanel.add(costRemoveButton, gridBagConstraints);

costValuesPanel.setLayout(new java.awt.GridBagLayout());

costLabel.setText("Cost");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
costValuesPanel.add(costLabel, gridBagConstraints);

costTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
costValuesPanel.add(costTextField, gridBagConstraints);

costChangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
costValuesPanel.add(costChangeTextField, gridBagConstraints);

costChangeLabel.setText("Cost Change");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
costValuesPanel.add(costChangeLabel, gridBagConstraints);

costAttackerMotiveLabel.setText("Attacker Motive");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 6;
costValuesPanel.add(costAttackerMotiveLabel, gridBagConstraints);

costAttackerMotiveTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 7;
costValuesPanel.add(costAttackerMotiveTextField, gridBagConstraints);
```

```
        costAttackerValueChangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 7;
        costValuesPanel.add(costAttackerValueChangeTextField, gridBagConstraints);

        costAttackerValueChangeLabel.setText("Attacker Value Change");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 6;
        costValuesPanel.add(costAttackerValueChangeLabel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        costListPanel.add(costValuesPanel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(20, 0, 20, 0);
        assetPanel.add(costListPanel, gridBagConstraints);

        assetScrollPane.setViewportView(assetPanel);

        add(assetScrollPane);

    }
//the following listeners set a text field based on a combo box selection
    private void assetNameComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        nameTextField.setText(assetNameComboBox.getSelectedItem().toString());
    }

    private void aclGroupComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclGroupComboBox.getSelectedItem() != null)
        {
            aclGroupFinalTextField.setText(
                    aclGroupComboBox.getSelectedItem().toString());
        }
    }

    private void costGroupComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(costGroupComboBox.getSelectedItem() != null)
        {
            costGroupFinalTextField.setText(
                    costGroupComboBox.getSelectedItem().toString());
        }
    }

    private void costUserComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(costUserComboBox.getSelectedItem() != null)
        {
            costUserFinalTextField.setText(
                    costUserComboBox.getSelectedItem().toString());
        }
    }

    private void integrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(integrityComboBox.getSelectedItem() != null)
        {
```

```
            integrityFinalTextField.setText(
                        integrityComboBox.getSelectedItem().toString());
        }
    }

    private void secrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(secrecyComboBox.getSelectedItem() != null)
        {
        secrecyFinalTextField.setText(
                        secrecyComboBox.getSelectedItem().toString());
        }
    }

    private void aclUserComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclUserComboBox.getSelectedItem() != null)
        {
            aclUserFinalTextField.setText(
                        aclUserComboBox.getSelectedItem().toString());
        }
    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
    private void costRemoveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if(costList.getSelectedIndex() >= 0)
        {
            costListTotalListVec.remove(costList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)costList.getModel();
            listModel.remove(costList.getSelectedIndex());
            costList = new JList(listModel);
            costList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            costScrollPane.setViewportView(costList);
        }
    }

    private void costAddButtonActionPerformed(java.awt.event.ActionEvent evt) {
        mCostList = new CostList();
        Object tempObject = new Object();
        CostList tempCostList = new CostList();
        mCostList.setUser(costUserFinalTextField.getText());
        mCostList.setGroup(costGroupFinalTextField.getText());
        mCostList.setRead(costReadComboBox.getSelectedItem().toString());
        mCostList.setWrite(costWriteComboBox.getSelectedItem().toString());
        mCostList.setControl(costControlComboBox.getSelectedItem().toString());
        mCostList.setExecute(costExecuteComboBox.getSelectedItem().toString());
        mCostList.setCost(costTextField.getText());
        mCostList.setCostChange(costChangeTextField.getText());
        mCostList.setAttackerMotive(costAttackerMotiveTextField.getText());
        mCostList.setAttackerValueChange(
                            costAttackerValueChangeTextField.getText());
        costListTotalListVec.addElement(mCostList);

        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < costListTotalListVec.size(); i++)
        {
            tempObject = costListTotalListVec.get(i);
            tempCostList = (CostList)tempObject;
            listModel.addElement(tempCostList.getCostList());
        }
        costList = null;
        costList = new JList(listModel);
        costList.setSelectionMode(
```

```
                javax.swing.ListSelectionModel.SINGLE_SELECTION);
    costScrollPane.setViewportView(costList);
    costList_added = true;
}

private void aclRemoveButtonActionPerformed(java.awt.event.ActionEvent evt) {
    if(aclList.getSelectedIndex() >= 0)
    {
        aclTotalAccessListVec.remove(aclList.getSelectedIndex());
        DefaultListModel listModel = new DefaultListModel();
        listModel = (DefaultListModel)aclList.getModel();
        listModel.remove(aclList.getSelectedIndex());
        aclList = new JList(listModel);
        aclList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        aclScrollPane.setViewportView(aclList);
    }
}

private void aclAddButtonActionPerformed(java.awt.event.ActionEvent evt) {
    mACL = new AccessControlList();
    Object tempObject = new Object();
    AccessControlList tempACL = new AccessControlList();
    mACL.setUser(aclUserFinalTextField.getText());
    mACL.setGroup(aclGroupFinalTextField.getText());
    mACL.setRead(aclReadComboBox.getSelectedItem().toString());
    mACL.setWrite(aclWriteComboBox.getSelectedItem().toString());
    mACL.setControl(aclControlComboBox.getSelectedItem().toString());
    mACL.setExecute(aclExecuteComboBox.getSelectedItem().toString());
    aclTotalAccessListVec.addElement(mACL);
    DefaultListModel listModel = new DefaultListModel();
    for(int i = 0; i < aclTotalAccessListVec.size(); i++)
    {
        tempObject = aclTotalAccessListVec.get(i);
        tempACL = (AccessControlList)tempObject;
        listModel.addElement(tempACL.getACL());
    }
    aclList = null;
    aclList = new JList(listModel);
    aclList.setSelectionMode(
                javax.swing.ListSelectionModel.SINGLE_SELECTION);
    aclScrollPane.setViewportView(aclList);
    acl_added = true;
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateDynamicSourceLists() takes datapath and startupScenario as
//parameters in ScenarioDefinitionTool.java and is used to add dynamic
//content to combo boxes and list boxes.
//First the scenairo is used to get the added reusable sets
//Second any combo boxes to be used are cleared and reinitialized
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
```

```
{
  //uses the Scenario passed to it to set the dynamic values of
  //dependent sets
  //User
  //DACGroup
  //Secrecy
  //Integrity
  mScenario = aScenario;
  Object tempObj = new Object();
  java.util.Vector tempVec = new java.util.Vector();
  ScenarioElementSet tempSet = new ScenarioElementSet();
  //User
  tempObj = aScenario.scenarioManager.get(11);
  tempVec = (Vector)tempObj;
  aclUserComboBox.removeAllItems();
  costUserComboBox.removeAllItems();
  aclUserComboBox.addItem("*");
  aclUserComboBox.addItem("$user");
  costUserComboBox.addItem("*");
  costUserComboBox.addItem("$user");
  for(int i = 0; i < tempVec.size(); i++)
  {
    tempObj = tempVec.get(i);
    try
    {
      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempObj.toString()));
    }
    catch(IOException ioException)
    {
      JOptionPane.showMessageDialog(this,
          "Exception during input stream creation", "Exception",
          JOptionPane.ERROR_MESSAGE);
    }
    try
    {
      tempSet = (ScenarioElementSet)input.readObject();
    }
    catch(ClassNotFoundException classnotfoundException)
    {
      JOptionPane.showMessageDialog(this,
          "Exception during file read - the object was not found",
          "Exception", JOptionPane.ERROR_MESSAGE);
    }
    catch(IOException ioException)
    {
      JOptionPane.showMessageDialog(this, "Exception during file read",
                "Exception", JOptionPane.ERROR_MESSAGE);
    }
    try
    {
      input.close();
    }
    catch(IOException ioException)
    {
      JOptionPane.showMessageDialog(this, "Exception during file close",
                  "Exception", JOptionPane.ERROR_MESSAGE);
    }
    for(int j = 0; j < tempSet.elementContainer.size(); j++)
    {
      User tempUser = new User();
      tempObj = tempSet.elementContainer.get(j);
```

```
                    tempUser = (User)tempObj;
                    aclUserComboBox.addItem(tempUser.getNameTextField());
                    costUserComboBox.addItem(tempUser.getNameTextField());
                 }
              }
              //DACGroup
              tempObj = aScenario.scenarioManager.get(4);
              tempVec = (Vector)tempObj;
              aclGroupComboBox.removeAllItems();
              costGroupComboBox.removeAllItems();
              aclGroupComboBox.addItem("*");
              aclGroupComboBox.addItem("$defaultGroup");
              costGroupComboBox.addItem("*");
              costGroupComboBox.addItem("$defaultGroup");
              for(int i = 0; i < tempVec.size(); i++)
              {
                 tempObj = tempVec.get(i);
                 try
                 {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempObj.toString()));
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showM essageDialog(this,
                          "Exception during input stream creation", "Exception",
                          JOptionPane.ERROR_MESSAGE);
                 }
                 try
                 {
                    tempSet = (ScenarioElementSet)input.readObject();
                 }
                 catch(ClassNotFoundException classnotfoundException)
                 {
                    JOptionPane.showMessageDialog(this,
                          "Exception during file read - the object was not found",
                          "Exception", JOptionPane.ERROR_MESSAGE);
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                 }
                 try
                 {
                    input.close();
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                 }
                 for(int j = 0; j < tempSet.elementContainer.size(); j++)
                 {
                    DACGroup tempDAC = new DACGroup();
                    tempObj = tempSet.elementContainer.get(j);
                    tempDAC = (DACGroup)tempObj;
                    aclGroupComboBox.addItem(tempDAC.getNameTextField());
                    costGroupComboBox.addItem(tempDAC.getNameTextField());
                 }
              }
              //Secrecy
```

```
                tempObj = aScenario.scenarioManager.get(9);
                tempVec = (Vector)tempObj;
                secrecyComboBox.removeAllItems();
                secrecyComboBox.addItem("$secrecy_clearance");
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   for(int j = 0; j < tempSet.elementContainer.size(); j++)
                   {
                      Secrecy tempSecrecy = new Secrecy();
                      tempObj = tempSet.elementContainer.get(j);
                      tempSecrecy = (Secrecy)tempObj;
                      secrecyComboBox.addItem(tempSecrecy.getNameTextField());
                   }
                }
                //Integrity
                tempObj = aScenario.scenarioManager.get(6);
                tempVec = (Vector)tempObj;
                integrityComboBox.removeAllItems();
                integrityComboBox.addItem("$integrity_clearance");
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempObj.toString()));
```

```
        }
        catch(IOException ioException)
        {
           JOptionPane.showMessageDialog(this,
                    "Exception during input stream creation", "Exception",
                    JOptionPane.ERROR_MESSAGE);
        }
        try
        {
           tempSet = (ScenarioElementSet)input.readObject();
        }
        catch(ClassNotFoundException classnotfoundException)
        {
           JOptionPane.showMessageDialog(this,
                    "Exception during file read - the object was not found",
                    "Exception", JOptionPane.ERROR_MESSAGE);
        }
        catch(IOException ioException)
        {
           JOptionPane.showMessageDialog(this, "Exception during file read",
                            "Exception", JOptionPane.ERROR_MESSAGE);
        }
        try
        {
           input.close();
        }
        catch(IOException ioException)
        {
           JOptionPane.showMessageDialog(this, "Exception during file close",
                               "Exception", JOptionPane.ERROR_MESSAGE);
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
           Integrity tempIntegrity = new Integrity();
           tempObj = tempSet.elementContainer.get(j);
           tempIntegrity = (Integrity)tempObj;
           integrityComboBox.addItem(tempIntegrity.getNameTextField());
        }
     }
  }

  public void Load()
  {
  }

  public void Save()
  {
  }

  //the following name_toString() methods below are used to create the
  //nested lists that appear in the Asset descriptor of an SDF.
  //These smaller name_toString() methods are called from the primary
  //asset toString() method.

  //creates the ACL list formatted output
  private String aclList_toString()
  {
     Object tempObject = new Object();
     AccessControlList tempACL = new AccessControlList();
     String aclString = new String();
     if(acl_added)
     {
```

```
      for(int i = 0; i < aclTotalAccessListVec.size(); i++)
      {
         aclString += "\tAccessList:\n";
         tempObject = aclTotalAccessListVec.get(i);
         tempACL = (AccessControlList)tempObject;
         aclString +=
         "\t\tAccess: " +
         tempACL.getUser() + "." + tempACL.getGroup()+
         " :end\n\t\tAccessMode: " +
         tempACL.getRead() + tempACL.getWrite() +
         tempACL.getControl() + tempACL.getExecute() + " :end\n"+
         "\t:end  //of AccessList\n\n";
      }
   }
   return aclString;
}

//creates the cost list formatted output
private String costList_toString()
{
   Object tempObject = new Object();
   CostList tempCostList = new CostList();
   String  costListString = new String();
   if(costList_added)
   {
      for(int i = 0; i < costListTotalListVec.size(); i++)
      {
         costListString += "\tCostList:\n\n";
         tempObject = costListTotalListVec.get(i);
         tempCostList = (CostList)tempObject;
         costListString +=
         "\t\tAccess: " +
         tempCostList.getUser() + "." + tempCostList.getGroup()+
         " :end\n"+
         "\t\tAccessMode: " +
         tempCostList.getRead() + tempCostList.getWrite() +
         tempCostList.getControl() + tempCostList.getExecute() +
         " :end\n\n"+
         "\t\t//The actual cost of this violation\n"+
         "\t\tCost: " + tempCostList.getCost() + " :end\n\n"+
         "\t\t//Integer factor from -10 to +10 determining how much"+
         " the value\n\t\t//changes per month.  -10 is max drop per"+
         " month, 0 is no change\n\t\t//per month, +10 is max"+
         " increase per month.\n"+
         "\t\tCostChange: "+tempCostList.getCostChange()+" :end\n\n"+
         "\t\t//How motivated the attacker is to violate this access"+
         " vector.\n"+
         "\t\tAttackerMotive: " + tempCostList.getAttackerMotive() +
         " :end\n\n"+
         "\t\t//Integer factor from -10 to +10 determining how much"+
         " the value\n\t\t//changes per month.  -10 is max drop per"+
         " month, 0 is no change\n\t\t//per month, +10 is max"+
         " increase per month.\n"+
         "\t\tAttackerValueChange: "+
         tempCostList.getAttackerValueChange()+" :end\n\n"+
         "\t:end    //of CostList\n\n";
      }
   }
   return costListString;
}

//Each descriptor form has a toString() method that is used by build() in
```

```
//the SDT to create the SDF.

//creates the formatted output for an asset descriptor
public String toString(String aText)
{
   String outputString = new String();
   outputString += "Asset:\n"+
   "\t//Asset describes an asset that can exist\n\t//on one or more"+
   " components at the start of a scenario.\n\t//Alternately, assets"+
   " can be created dynamically by virtual\n\t//users based on the user"+
   " goals.  In the latter case, asset\n\t//definitions can include"+
   " symbolic values that are resolved\n\t//based on the attributes"+
   " of the user who creates the asset.\n\n\t//A text string naming this"+
   " asset.  The name can include any of the\n\t//following symbols, "+
   "which are resolved based on the attributes of the user\n\t//who"+
   " creates the asset: $user, $default_group, $secrecy_clearance,"+
   " \n\t//$integrity_clearance.\n"+
   "\tName: "+nameTextField.getText()+" :end\n\n"+
   "\t//A boolean stating whether this asset exists at start time."+
   "  If false\n\t//then it must be created.\n"+
   "\tIsInstantiated: "+
   String.valueOf(isInstaniatedCheckBox.isSelected())+" :end\n\n"+
   "\t//A boolean value that specifies wether this asset has a DAC on it"+
   " or not\n"+
   "\tHasDAC: "+String.valueOf(hasDACCheckBox.isSelected())+
   " :end\n\n"+
   "\t//Either the tag name from a \"Secrecy label\", or\n\t//the"+
   " symbolic: $secrecy_clearance to reflect the secrecy clearance\n\t//"+
   "of the user that creates the asset.\n"+
   "\tSecrecy: "+secrecyFinalTextField.getText()+" :end\n\n"+
   "\t//Either the tag name from a \"Integrity label\", or\n\t//the"+
   " symbolic: $integrity_clearance to reflect the integrity clearance\n"+
   "\t//of the user that creates the asset.\n"+
   "\tIntegrity: "+integrityFinalTextField.getText()+" :end\n\n"+
   "\t//Denial of Service (DOS) motive is an abstract number representing"+
   "how\n\t//motivated the attackers are to complete a DOS attack on"+
   " this asset\n"+
   "\tDOSMotive: "+dosMotiveTextField.getText()+" :end\n\n"+
   "\t//Availability is the penalty for not having this asset available\n"+
   "\tAvailabilityPenalty: "+availabilityPenaltyTextField.getText()+
   " :end\n\n"+
   "\t//The following example that you can either type a %user.group%"+
   " intending\n\t//specific user access while the user is working"+
   " within a specific group, or you\n\t//can use the wild card"+
   " designator in place of either the user or the group\n\t//For"+
   " example to intend access to a whole group the syntax is:"+
   " %*.group%.\n\n\t//Alternately, to intend the specific user"+
   " regardless of the current group use the\n\t//syntax: %user.*%.  "+
   "Also, instead of a specific user Tag Name, the user value"+
   " can\n\t//be $user, which is replaced by the Tag Name of the user"+
   " who creates the asset.\n\t//Similarly, the group value can be"+
   "  $default_group which is replaced by the\n\t//default group"+
   " associated with the creating user.\n\n\t//Most specific dominates "+
   "when resolving\n\n\t//This list is intended access.\n\t//Read, write,"+
   " control, execute values are yes, no, or don't care\n\t//Access"+
   " List\t//If HasDAC == true\n\t//(*|%name%|$user).(*|%group%|$def"+
   "ault_group) %read% %write%\n\t//%control% %execute%\n\t//Access"+
   " list entry.  Available modes are Y = Yes, N = No and X = Don't"+
   " care\n\t//The asset can have any number of AccessList blocks\n"+
   aclList_toString()+
   "\t//Cost list defines the penalties incurred if a particular access"+
   " violation\n\t//occurs.  In such an event, the \"Cost List\" is"+
```

```
    " referenced to\n\t//determine the cost based on which one of these"+
    " groups or people access\n\t//the asset.\n\n\t//Name or group can"+
    " be expressed as teh wildcard charater: \"*\".\n\t//Mode is read,"+
    " write, control, execute, or any\n\t//Cost is the abstract cost of "+
    "a violation\n\t//Attacker motive is an abstract number representing"+
    "the motive for the\n\t//attacker.\n\n\t//Mode for the cost list "+
    "is, from left to right, read/write/control/execute\n\t//where 'X'"+
    " triggers the property while '-' does not.  for example,\n\t//XX--"+
    " would mean that if read or write access is violated then "+
    "the\n\t//penalty amount in cost field would be triggered; a --X- "+
    "would mean\n\t//that only if the control of the asset is violated"+
    " would the cost be\n\t//triggered\n\n"+
    costList_toString()+
    ":end    //of Asset\n\n";
    return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
    return nameTextField.getText();
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
    Object tempObject = new Object();
    AccessControlList tempACL = new AccessControlList();
    CostList tempCostList = new CostList();
    DefaultListModel listModel = new DefaultListModel();
    for(int i = 0; i < costListTotalListVec.size(); i++)
    {
        tempObject = costListTotalListVec.get(i);
        tempCostList = (CostList)tempObject;
        listModel.addElement(tempCostList.getCostList());
    }
    costList = null;
    costList = new JList(listModel);
    costList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
    costScrollPane.setViewportView(costList);
    costList_added = true;

    tempObject = new Object();
    listModel = new DefaultListModel();
    for(int i = 0; i < aclTotalAccessListVec.size(); i++)
    {
        tempObject = aclTotalAccessListVec.get(i);
        tempACL = (AccessControlList)tempObject;
        listModel.addElement(tempACL.getACL());
    }
    aclList = null;
    aclList = new JList(listModel);
    aclList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
    aclScrollPane.setViewportView(aclList);
    acl_added = true;
}
```

```
//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
   costAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         costAddButtonActionPerformed(evt);
      }
   });

   costRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         costRemoveButtonActionPerformed(evt);
      }
   });

   aclAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclAddButtonActionPerformed(evt);
      }
   });

   aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclRemoveButtonActionPerformed(evt);
      }
   });
}

public void reInitListeners(String aPath, Scenario aScenario)
{
   populateDynamicSourceLists(aPath, aScenario);

   aclUserComboBox.addActionListener(new java.awt.event.ActionListener()
   {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclUserComboBoxActionPerformed(evt);
      }
   });

   aclGroupComboBox.addActionListener(new java.awt.event.ActionListener()
   {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclGroupComboBoxActionPerformed(evt);
      }
   });

   secrecyComboBox.addActionListener(new java.awt.event.ActionListener()
   {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         secrecyComboBoxActionPerformed(evt);
      }
   });

   integrityComboBox.addActionListener(new java.awt.event.ActionListener()
   {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         integrityComboBoxActionPerformed(evt);
      }
   });
```

```
    costUserComboBox.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            costUserComboBoxActionPerformed(evt);
        }
    });

    costGroupComboBox.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            costGroupComboBoxActionPerformed(evt);
        }
    });

    assetNameComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            assetNameComboBoxActionPerformed(evt);
        }
    });

}

// Variables declaration - do not modify
private javax.swing.JButton aclAddButton;
private javax.swing.JComboBox aclControlComboBox;
private javax.swing.JLabel aclControlLabel;
private javax.swing.JComboBox aclExecuteComboBox;
private javax.swing.JLabel aclExecuteLabel;
private javax.swing.JComboBox aclGroupComboBox;
private javax.swing.JTextField aclGroupFinalTextField;
private javax.swing.JLabel aclGroupLabel;
private javax.swing.JList aclList;
private javax.swing.JPanel aclPanel;
private javax.swing.JComboBox aclReadComboBox;
private javax.swing.JLabel aclReadLabel;
private javax.swing.JButton aclRemoveButton;
private javax.swing.JScrollPane aclScrollPane;
private javax.swing.JComboBox aclUserComboBox;
private javax.swing.JTextField aclUserFinalTextField;
private javax.swing.JLabel aclUserLabel;
private javax.swing.JComboBox aclWriteComboBox;
private javax.swing.JLabel aclWriteLabel;
private javax.swing.JComboBox assetNameComboBox;
private javax.swing.JPanel assetPanel;
private javax.swing.JScrollPane assetScrollPane;
private javax.swing.JLabel availabilityPenaltyLabel;
private javax.swing.JTextField availabilityPenaltyTextField;
private javax.swing.JPanel boolPanel;
private javax.swing.JButton costAddButton;
private javax.swing.JLabel costAttackerMotiveLabel;
private javax.swing.JTextField costAttackerMotiveTextField;
private javax.swing.JLabel costAttackerValueChangeLabel;
private javax.swing.JTextField costAttackerValueChangeTextField;
private javax.swing.JLabel costChangeLabel;
private javax.swing.JTextField costChangeTextField;
private javax.swing.JComboBox costControlComboBox;
private javax.swing.JLabel costControlLabel;
private javax.swing.JComboBox costExecuteComboBox;
private javax.swing.JLabel costExecuteLabel;
private javax.swing.JComboBox costGroupComboBox;
private javax.swing.JTextField costGroupFinalTextField;
```

```
    private javax.swing.JLabel costGroupLabel;
    private javax.swing.JLabel costLabel;
    private javax.swing.JList costList;
    private javax.swing.JPanel costListPanel;
    private javax.swing.JComboBox costReadComboBox;
    private javax.swing.JLabel costReadLabel;
    private javax.swing.JButton costRemoveButton;
    private javax.swing.JScrollPane costScrollPane;
    private javax.swing.JTextField costTextField;
    private javax.swing.JComboBox costUserComboBox;
    private javax.swing.JTextField costUserFinalTextField;
    private javax.swing.JLabel costUserLabel;
    private javax.swing.JPanel costValuesPanel;
    private javax.swing.JComboBox costWriteComboBox;
    private javax.swing.JLabel costWriteLabel;
    private javax.swing.JLabel dosMotiveLabel;
    private javax.swing.JTextField dosMotiveTextField;
    private javax.swing.JCheckBox hasDACCheckBox;
    private javax.swing.JComboBox integrityComboBox;
    private javax.swing.JTextField integrityFinalTextField;
    private javax.swing.JLabel integrityLabel;
    private javax.swing.JCheckBox isInstaniatedCheckBox;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JComboBox secrecyComboBox;
    private javax.swing.JTextField secrecyFinalTextField;
    private javax.swing.JLabel secrecyLabel;
    private javax.swing.JPanel textAndComboPanel;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private Scenario mScenario;
    private boolean acl_added;
    private AccessControlList mACL;
    private Vector aclTotalAccessListVec;
    private boolean costList_added;
    private CostList mCostList;
    private Vector costListTotalListVec;
}
```

C. THE SOURCE OF: ASSET GOAL

```
/*
 * AssetGoal.java
 *
 * Created on January 23, 2004, 9:47 AM
 */

package CCSDT;
import javax.swing.*;
import java.io.*;
/**
 *
 * @author  Ken
 *This class is the graphical representation of an AssetGoal descriptor.  This
 *is where the AssetGoal form is managed and the AssetGoal toString() resides.
 */
public class AssetGoal extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form AssetGoal */
    public AssetGoal()
    {
```

```
    initComponents();
    totalAssetGoalAssetVec = new java.util.Vector();
    asset_added = false;
    assetControlComboBox.addItem("Y");
    assetControlComboBox.addItem("N");
    assetControlComboBox.addItem("X");
    assetExecuteComboBox.addItem("Y");
    assetExecuteComboBox.addItem("N");
    assetExecuteComboBox.addItem("X");
    assetReadComboBox.addItem("Y");
    assetReadComboBox.addItem("N");
    assetReadComboBox.addItem("X");
    assetWriteComboBox.addItem("Y");
    assetWriteComboBox.addItem("N");
    assetWriteComboBox.addItem("X");
}

private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;

    assetGoalScrollPane = new javax.swing.JScrollPane();
    assetGoalPanel = new javax.swing.JPanel();
    nameLabel = new javax.swing.JLabel();
    nameTextField = new javax.swing.JTextField();
    descriptionScrollPane = new javax.swing.JScrollPane();
    descriptionTextArea = new javax.swing.JTextArea();
    descriptionLabel = new javax.swing.JLabel();
    sharedCheckBox = new javax.swing.JCheckBox();
    costPenaltyLabel = new javax.swing.JLabel();
    costPenaltyChangeL abel = new javax.swing.JLabel();
    costPenaltyTextField = new javax.swing.JTextField();
    costPenaltyChangeTextField = new javax.swing.JTextField();
    assetPanel = new javax.swing.JPanel();
    assetLabel = new javax.swing.JLabel();
    assetReadLabel = new javax.swing.JLabel();
    assetWriteLabel = new javax.swing.JLabel();
    assetControlLabel = new javax.swing.JLabel();
    assetExecuteLabel = new javax.swing.JLabel();
    assetComboBox = new javax.swing.JComboBox();
    assetFinalTextField = new javax.swing.JTextField();
    assetReadComboBox = new javax.swing.JComboBox();
    assetWriteComboBox = new javax.swing.JComboBox();
    assetControlComboBox = new javax.swing.JComboBox();
    assetExecuteComboBox = new javax.swing.JComboBox();
    assetAddButton = new javax.swing.JButton();
    assetScrollPane = new javax.swing.JScrollPane();
    assetList = new javax.swing.JList();
    assetRemoveButton = new javax.swing.JButton();

    setLayout(new java.awt.GridLayout(1, 0));

    assetGoalPanel.setLayout(new java.awt.GridBagLayout());

    nameLabel.setLabelFor(nameTextField);
    nameLabel.setText("Asset Goal Name:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    assetGoalPanel.add(nameLabel, gridBagConstraints);

    nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
assetGoalPanel.add(nameTextField, gridBagConstraints);

descriptionScrollPane.setPreferredSize(new java.awt.Dimension(425, 60));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setWrapStyleWord(true);
descriptionTextArea.setMinimumSize(new java.awt.Dimension(0, 0));
descriptionTextArea.setPreferredSize(new java.awt.Dimension(425, 1000));
descriptionScrollPane.setViewportView(descriptionTextArea);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
assetGoalPanel.add(descriptionScrollPane, gridBagConstraints);

descriptionLabel.setText("Description:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
gridBagConstraints.insets = new java.awt.Insets(4, 0, 0, 0);
assetGoalPanel.add(descriptionLabel, gridBagConstraints);

sharedCheckBox.setText("This goal is shared among multiple users");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
assetGoalPanel.add(sharedCheckBox, gridBagConstraints);

costPenaltyLabel.setText("Availability Penalty:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
assetGoalPanel.add(costPenaltyLabel, gridBagConstraints);

costPenaltyChangeLabel.setText("Availabiltiy Penalty Change:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
assetGoalPanel.add(costPenaltyChangeLabel, gridBagConstraints);

costPenaltyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
assetGoalPanel.add(costPenaltyTextField, gridBagConstraints);

costPenaltyChangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
assetGoalPanel.add(costPenaltyChangeTextField, gridBagConstraints);

assetPanel.setLayout(new java.awt.GridBagLayout());

assetPanel.setBorder(new javax.swing.border.TitledBorder("Assets Defined for This Goal"));
```

```
assetLabel.setText("Asset");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
assetPanel.add(assetLabel, gridBagConstraints);

assetReadLabel.setText("Read");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
assetPanel.add(assetReadLabel, gridBagConstraints);

assetWriteLabel.setText("Write");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
assetPanel.add(assetWriteLabel, gridBagConstraints);

assetControlLabel.setText("Control");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 3;
assetPanel.add(assetControlLabel, gridBagConstraints);

assetExecuteLabel.setText("Execute");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 30, 0, 0);
assetPanel.add(assetExecuteLabel, gridBagConstraints);

assetComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
assetComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      assetComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
assetPanel.add(assetComboBox, gridBagConstraints);

assetFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
assetPanel.add(assetFinalTextField, gridBagConstraints);

assetReadComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
assetPanel.add(assetReadComboBox, gridBagConstraints);

assetWriteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
assetPanel.add(assetWriteComboBox, gridBagConstraints);
```

```
        assetControlComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 4;
        gridBagConstraints.gridy = 4;
        assetPanel.add(assetControlComboBox, gridBagConstraints);

        assetExecuteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 5;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        assetPanel.add(assetExecuteComboBox, gridBagConstraints);

        assetAddButton.setText("Add");
        assetAddButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                assetAddButtonActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        assetPanel.add(assetAddButton, gridBagConstraints);

        assetScrollPane.setPreferredSize(new java.awt.Dimension(700, 150));
        assetScrollPane.setViewportView(assetList);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 7;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        assetPanel.add(assetScrollPane, gridBagConstraints);

        assetRemoveButton.setText("Remove");
        assetRemoveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                assetRemoveButtonActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        assetPanel.add(assetRemoveButton, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(20, 0, 20, 0);
        assetGoalPanel.add(assetPanel, gridBagConstraints);

        assetGoalScrollPane.setViewportView(assetGoalPanel);

        add(assetGoalScrollPane);

    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
```

```
    private void assetRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
       if(assetList.getSelectedIndex() >= 0)
       {
          totalAssetGoalAssetVec.remove(assetList.getSelectedIndex());
          DefaultListModel listModel = new DefaultListModel();
          listModel = (DefaultListModel)assetList.getModel();
          listModel.remove(assetList.getSelectedIndex());
          assetList = new JList(listModel);
          assetList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
          assetScrollPane.setViewportView(assetList);
       }
    }

    private void assetAddButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
       mAGA = new AssetGoalAsset();
       Object tempObject = new Object();
       AssetGoalAsset tempAGA = new AssetGoalAsset();
       mAGA.setAsset(assetFinalTextField.getText());
       mAGA.setRead(assetReadComboBox.getSelectedItem().toString());
       mAGA.setWrite(assetWriteComboBox.getSelectedItem().toString());
       mAGA.setControl(assetControlComboBox.getSelectedItem().toString());
       mAGA.setExecute(assetExecuteComboBox.getSelectedItem().toString());
       totalAssetGoalAssetVec.addElement(mAGA);
       DefaultListModel listModel = new DefaultListModel();
       for(int i = 0; i < totalAssetGoalAssetVec.size(); i++)
       {
          tempObject = totalAssetGoalAssetVec.get(i);
          tempAGA = (AssetGoalAsset)tempObject;
          listModel.addElement(tempAGA.getAssetGoalAssetString());
       }
       assetList = null;
       assetList = new JList(listModel);
       assetList.setSelectionMode(
                      javax.swing.ListSelectionModel.SINGLE_SELECTION);
       assetScrollPane.setViewportView(assetList);
       asset_added = true;
    }
//the following listener sets a text field based on a combo box selection
    private void assetComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
       if(assetComboBox.getSelectedItem() != null)
       {
          assetFinalTextField.setText(assetComboBox.getSelectedItem().
                                                toString());
       }
    }

    //Interface combo box and list box content comes from one of two sources
    //the content is either from ini files or from reusable sets that have
    //already been added to the scenario in development.
    //If the content come from ini files it is designated static.  This is
    //because the content of ini files does not change.
    //If the content comes from reusable sets that have been added to the
    //scenario in development it is designated dynamic.  This is because
    //the content may be empty, of any lenght and change frequently

    //populateDynamicSourceLists() takes datapath and startupScenario as
    //parameters in ScenarioDefinitionTool.java and is used to add dynamic
    //content to combo boxes and list boxes.
    //First the scenairo is used to get the added reusable sets
    //Second any combo boxes to be used are cleared and reinitialized
```

```
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
{
   //uses the Scenario passed to it to set the dynamic values of
   //dependent sets
   //Asset
   mScenario = aScenario;
   Object tempObj = new Object();
   java.util.Vector tempVec = new java.util.Vector();
   ScenarioElementSet tempSet = new ScenarioElementSet();

   //Asset
   assetComboBox.removeAllItems();
   tempObj = aScenario.scenarioManager.get(0);
   tempVec = (java.util.Vector)tempObj;
   for(int i = 0; i < tempVec.size(); i++)
   {
      tempObj = tempVec.get(i);
      try
      {
         input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Asset/"+tempObj.toString()));
      }
      catch(IOException ioException)
      {
         JOptionPane.showMessageDialog(this,
                  "Exception during input stream creation", "Exception",
                  JOptionPane.ERROR_MESSAGE);
      }
      try
      {
         tempSet = (ScenarioElementSet)input.readObject();
      }
      catch(ClassNotFoundException classnotfoundException)
      {
         JOptionPane.showMessageDialog(this,
                  "Exception during file read - the object was not found",
                  "Exception", JOptionPane.ERROR_MESSAGE);
      }
      catch(IOException ioException)
      {
         JOptionPane.showMessageDialog(this, "Exception during file read",
                        "Exception", JOptionPane.ERROR_MESSAGE);
      }
      try
      {
         input.close();
      }
      catch(IOException ioException)
      {
         JOptionPane.showMessageDialog(this, "Exception during file close",
                           "Exception", JOptionPane.ERROR_MESSAGE);
      }
      for(int j = 0; j < tempSet.elementContainer.size(); j++)
      {
         Asset tempAsset = new Asset();
         tempObj = tempSet.elementContainer.get(j);
         tempAsset = (Asset)tempObj;
         assetComboBox.addItem(tempAsset.getNameTextField());
      }
```

```
   }
}

public void Load()
{
}

public void Save()
{
}

//the following nametoString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller nametoString() methods are called from the primary
//asset toString() method.
private String assetGoalAssetToString()
{
   Object tempObject = new Object();
   AssetGoalAsset tempAssetGoalAsset = new AssetGoalAsset();
    String assetGoalAssetString = new String();
   if(asset_added)
   {
      for(int i = 0; i < totalAssetGoalAssetVec.size(); i++)
      {
         assetGoalAssetString += "\tAsset:\n";
         tempObject = totalAssetGoalAssetVec.get(i);
         tempAssetGoalAsset = (AssetGoalAsset)tempObject;
         assetGoalAssetString +=
         "\t\t//Name of the asset that must be accessed.  This may be "+
         "the name\n\t\t//of a Software program, e.g., if Mode is "+
         "execute.\n" +
         "\t\tName: "+tempAssetGoalAsset.getAsset()+" :end\n\n"+
         "\t\t//Mode in the form Read Write Control Execute with a Y, "+
         "N, X for\n\t\t//yes, no, don't care\n"+
         "\t\tAccessMode: "+tempAssetGoalAsset.getRead()+
         tempAssetGoalAsset.getWrite()+tempAssetGoalAsset.getControl()+
         tempAssetGoalAsset.getExecute()+" :end\n"+
         "\t:end    //of Asset\n\n";
      }
   }
   return assetGoalAssetString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an assetgoal descriptor
public String toString(String aText)
{
   String outputString = new String();
   outputString +=
   "AssetGoal:\n"+
   "\t//Tag name to use for this asset goal\n"+
   "\tName: "+nameTextField.getText()+" :end\n\n"+
   "\t//A description of the Asset Goal\n"+
   "\tDescription: "+descriptionTextArea.getText()+" :end\n\n"+
   "\t//If the shared field in the asset goal is set to true, then if "+
   "multiple users are\n\t//assigned this asset goal, then can only "+
   "succeed in this goal if all users assigned\n\t//this goal can "+
   "succeed.  They don't have to reach the asset(s) at the same time, "+
   "they\n\t//just can't succeed unless they all succeed (i.e. they all "+
   "have to be able to get\n\t//adequate access to their assets at some "+
```

```
   "point.)\n\n\t//If the shared field in the asset goal is not set to "+
   "true, then if the asset goal is\n\t//assigned to multiple users, "+
   "the multiple users are not dependent on the other\n\t//users "+
   "succeeding in order for their asset goal to succeed.\n"+
   "\tShared: "+String.valueOf(sharedCheckBox.isSelected())+" :end\n\n"+
   "\t//A list of assets and modes that define this asset goal\n\n\t//If "+
   "multiple asset blockes are listed within an asset goal then it "+
   "always means that\n\t//the user has to be able toa access those "+
   "assets at the same time.  There will\n\t//not be a way to list "+
   "multiple assets in an asset goal that doesn't mean\n\t//simultaneous "+
   "access is required.\n"+
   assetGoalAssetToString()+
   "\t//This is the penalty is the asset goal cannot be fully "+
   "accessed.\n\t//The units are dollars per month\n"+
   "\tAvailabilityCostPenalty: "+costPenaltyTextField.getText()+" :end\n\n"+
   "\t//Integer factor from -10 to +10 determining how much "+
   "the\n\t//AvailabilityCostPenalty changes per month.  -10 is max "+
   "drop\n\t//per month, 0 is no changer per month, +10 is max increase "+
   "per month.\n"+
   "\tAvailabilityCostPenaltyChange: "+
   costPenaltyChangeTextField.getText()+" :end\n\n"+
   ":end   //of AssetGoal\n\n";
   return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
   return nameTextField.getText();
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
   Object tempObject = new Object();
   AssetGoalAsset tempAGA = new AssetGoalAsset();
   DefaultListModel listModel = new DefaultListModel();
   for(int i = 0; i < totalAssetGoalAssetVec.size(); i++)
   {
      tempObject = t otalAssetGoalAssetVec.get(i);
      tempAGA = (AssetGoalAsset)tempObject;
      listModel.addElement(tempAGA.getAssetGoalAssetString());
   }
   assetList = null;
   assetList = new JList(listModel);
   assetList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   assetScrollPane.setViewportView(assetList);
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
   assetAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         assetAddButtonActionPerformed(evt);
```

```
        }
    });

    assetRemoveButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            assetRemoveButtonActionPerformed(evt);
        }
    });
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario)
{
    populateDynamicSourceLists(aPath, aScenario);

    assetComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
    assetComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            assetComboBoxActionPerformed(evt);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton assetAddButton;
private javax.swing.JComboBox assetComboBox;
private javax.swing.JComboBox assetControlComboBox;
private javax.swing.JLabel assetControlLabel;
private javax.swing.JComboBox assetExecuteComboBox;
private javax.swing.JLabel assetExecuteLabel;
private javax.swing.JTextField assetFinalTextField;
private javax.swing.JPanel assetGoalPanel;
private javax.swing.JScrollPane assetGoalScrollPane;
private javax.swing.JLabel assetLabel;
private javax.swing.JList assetList;
private javax.swing.JPanel assetPanel;
private javax.swing.JComboBox assetReadComboBox;
private javax.swing.JLabel assetReadLabel;
private javax.swing.JButton assetRemoveButton;
private javax.swing.JScrollPane assetScrollPane;
private javax.swing.JComboBox assetWriteComboBox;
private javax.swing.JLabel assetWriteLabel;
private javax.swing.JLabel costPenaltyChangeLabel;
private javax.swing.JTextField costPenaltyChangeTextField;
private javax.swing.JLabel costPenaltyLabel;
private javax.swing.JTextField costPenaltyTextField;
private javax.swing.JLabel descriptionLabel;
private javax.swing.JScrollPane descriptionScrollPane;
private javax.swing.JTextArea descriptionTextArea;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
private javax.swing.JCheckBox sharedCheckBox;
// End of variables declaration
transient private java.io.ObjectInputStream input;
private Scenario mScenario;
private AssetGoalAsset mAGA;
private java.util.Vector totalAssetGoalAssetVec;
private boolean asset_added;
}
```

D. THE SOURCE OF: ASSET GOAL ASSET

```
/*
 * AssetGoalAsset.java
 *
 * Created on February 15, 2004, 9:57 PM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is used as an intermediate container to store user choices for
 *an asset in asset goal.
 */
public class AssetGoalAsset implements java.io.Serializable
{

    /** Creates a new instance of AssetGoalAsset */
    public AssetGoalAsset()
    {
        asset = new String();
        read = new String();
        write = new String();
        control = new String();
        execute = new String();
    }

    public void setAsset(String aAsset)
    {
        asset = aAsset;
    }
    public void setRead(String aRead)
    {
        read = aRead;
    }
    public void setWrite(String aWrite)
    {
        write = aWrite;
    }
    public void setControl(String aControl)
    {
        control = aControl;
    }
    public void setExecute(String aExecute)
    {
        execute = aExecute;
    }
    public String getAsset()
    {
        return asset;
    }
    public String getRead()
    {
        return read;
    }
    public String getWrite()
    {
        return write;
    }
    public String getControl()
```

```
    {
        return control;
    }
    public String getExecute()
    {
        return execute;
    }
    public String getAssetGoalAssetString()
    {
        return "(ASSET)"+asset + "(READ)" + read + "(WRITE)" + write + "(CONTROL)" +
            control + "(EXECUTE)" + execute;
    }

    private String asset;
    private String read;
    private String write;
    private String control;
    private String execute;
}
```

E. THE SOURCE OF: CATALOG COMPONENT

```
/*
 * CatalogComponent.java
 *
 * Created on January 7, 2004, 12:57 PM
 */

package CCSDT;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.JOptionPane;

/**
 *
 * @author  kjohns
 *This class is the graphical representation of a CatalogComponent descriptor.
 *This is where the CatalogComponent form is managed and the CatalogComponent
 *toString() resides.
 */
public class CatalogComponent extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form CatalogComponent */
    public CatalogComponent()
    {
        initComponents();
        IsTemplate = true;
        software_added = false;
        mSoftware_Vector = new Vector();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        patchFreqbuttonGroup = new javax.swing.ButtonGroup();
        antiVirusbuttonGroup = new javax.swing.ButtonGroup();
        browserbuttonGroup = new javax.swing.ButtonGroup();
        emailbuttonGroup = new javax.swing.ButtonGroup();
        catalogComponentScrollPane = new javax.swing.JScrollPane();
        catalogComponentPanel = new javax.swing.JPanel();
```

```
comboBoxandTextPanel1 = new javax.swing.JPanel();
nameLabel = new javax.swing.JLabel();
nameTextField = new javax.swing.JTextField();
descriptionScrollPane = new javax.swing.JScrollPane();
descriptionTextArea = new javax.swing.JTextArea();
costTextField = new javax.swing.JTextField();
resaleTextField = new javax.swing.JTextField();
maintenanceTextField = new javax.swing.JTextField();
availabilityTextField = new javax.swing.JTextField();
availablityLabel = new javax.swing.JLabel();
maintenanceLabel = new javax.swing.JLabel();
resaleLabel = new javax.swing.JLabel();
costLabel = new javax.swing.JLabel();
descriptionLabel = new javax.swing.JLabel();
listsPanel = new javax.swing.JPanel();
softwareLabel = new javax.swing.JLabel();
softwareSourceScrollPane = new javax.swing.JScrollPane();
softwareSourceList = new javax.swing.JList();
softwareMoveButton = new javax.swing.JButton();
softwareFinalScrollPane = new javax.swing.JScrollPane();
softwareFinalList = new javax.swing.JList();
softwareRemoveButton = new javax.swing.JButton();
manyToOnePanel = new javax.swing.JPanel();
hardwareNameLabel = new javax.swing.JLabel();
opSysLabel = new javax.swingJLabel();
opSysSourceComboBox = new javax.swing.JComboBox();
hardwareSourceComboBox = new javax.swing.JComboBox();
boolPanel1 = new javax.swing.JPanel();
patchesPanel = new javax.swing.JPanel();
patchFreqAsReleasedRadioButton = new javax.swing.JRadioButton();
patchFreqRoutinelyRadioButton = new javax.swing.JRadioButton();
patchFreqAutoRadioButton = new javax.swing.JRadioButton();
avPanel = new javax.swing.JPanel();
antiVirusRegRadioButton = new javax.swing.JRadioButton();
antiVirusAutoRadioButton = new javax.swing.JRadioButton();
browserPanel = new javax.swing.JPanel();
browserLooseRadioButton = new javax.swing.JRadioButton();
browserNormalRadioButton = new javax.swing.JRadioButton();
browserStrictRadioButton = new javax.swing.JRadioButton();
emailPanel = new javax.swing.JPanel();
emailLooseRadioButton = new javax.swing.JRadioButton();
emailNormalRadioButton = new javax.swing.JRadioButton();
emailStrictRadioButton = new javax.swing.JRadioButton();
checkboxPanel = new javax.swing.JPanel();
assetProtectionCheckBox = new javax.swing.JCheckBox();
remoteAccessCheckBox = new javax.swing.JCheckBox();
pKICertsCheckBox = new javax.swing.JCheckBox();
oneTimePassCheckBox = new javax.swing.JCheckBox();
biometricsCheckBox = new javax.swing.JCheckBox();
tokenPKICheckBox = new javax.swing.JCheckBox();
clientPKICheckBox = new javax.swing.JCheckBox();
vPNClientCheckBox = new javax.swing.JCheckBox();
scanEmailCheckBox = new javax.swing.JCheckBox();
stripEmailCheckBox = new javax.swing.JCheckBox();
autoLogoutCheckBox = new javax.swing.JCheckBox();
userAdminNoMACCheckBox = new javax.swing.JCheckBox();
userAdminMACCheckBox = new javax.swing.JCheckBox();
adminOnlyInstallsCheckBox = new javax.swing.JCheckBox();
blockMediaCheckBox = new javax.swing.JCheckBox();
blockWriteCheckBox = new javax.swing.JCheckBox();

setLayout(new java.awt.GridLayout(1, 0));
```

```
setPreferredSize(new java.awt.Dimension(611, 1172));
catalogComponentScrollPane.setPreferredSize(new java.awt.Dimension(611, 1172));
catalogComponentPanel.setLayout(new java.awt.GridBagLayout());

catalogComponentPanel.setMinimumSize(new java.awt.Dimension(400, 340));
catalogComponentPanel.setPreferredSize(new java.awt.Dimension(650, 820));
comboBoxandTextPanel1.setLayout(new java.awt.GridBagLayout());

nameLabel.setText("Component Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel1.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(nameTextField, gridBagConstraints);

descriptionScrollPane.setPreferredSize(new java.awt.Dimension(425, 60));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setWrapStyleWord(true);
descriptionTextArea.setMinimumSize(new java.awt.Dimension(0, 0));
descriptionTextArea.setPreferredSize(new java.awt.Dimension(425, 1000));
descriptionScrollPane.setViewportView(descriptionTextArea);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(descriptionScrollPane, gridBagConstraints);

costTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(costTextField, gridBagConstraints);

resaleTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(resaleTextField, gridBagConstraints);

maintenanceTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(maintenanceTextField, gridBagConstraints);

availabilityTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel1.add(availabilityTextField, gridBagConstraints);

availablityLabel.setText("Percentage of Time Available:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel1.add(availablityLabel, gridBagConstraints);

maintenanceLabel.setText("Maintenance Cost:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel1.add(maintenanceLabel, gridBagConstraints);

resaleLabel.setText("Resale Value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel1.add(resaleLabel, gridBagConstraints);

costLabel.setText("Purchase Price:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel1.add(costLabel, gridBagConstraints);

descriptionLabel.setText("Component Description:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
gridBagConstraints.insets = new java.awt.Insets(4, 0, 0, 0);
comboBoxandTextPanel1.add(descriptionLabel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
catalogComponentPanel.add(comboBoxandTextPanel1, gridBagConstraints);

listsPanel.setLayout(new java.awt.GridBagLayout());

softwareLabel.setText("Software for this Device:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(softwareLabel, gridBagConstraints);

softwareSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
softwareSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
softwareSourceScrollPane.setViewportView(softwareSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(softwareSourceScrollPane, gridBagConstraints);
```

```
softwareMoveButton.setText(">>>>>");
softwareMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      softwareMoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
listsPanel.add(softwareMoveButton, gridBagConstraints);

softwareFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
softwareFinalScrollPane.setViewportView(softwareFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 1;
listsPanel.add(softwareFinalScrollPane, gridBagConstraints);

softwareRemoveButton.setText("Remove");
softwareRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      softwareRemoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 2;
listsPanel.add(softwareRemoveButton, gridBagConstraints);

manyToOnePanel.setLayout(new java.awt.GridBagLayout());

hardwareNameLabel.setText("Base Component for this Device:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(hardwareNameLabel, gridBagConstraints);

opSysLabel.setText("Operating System:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(opSysLabel, gridBagConstraints);

opSysSourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
manyToOnePanel.add(opSysSourceComboBox, gridBagConstraints);

hardwareSourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
manyToOnePanel.add(hardwareSourceComboBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
listsPanel.add(manyToOnePanel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
catalogComponentPanel.add(listsPanel, gridBagConstraints);

boolPanel1.setLayout(new java.awt.GridBagLayout());

boolPanel1.setBorder(new javax.swing.border.TitledBorder("Configuration Settings"));
patchesPanel.setLayout(new java.awt.GridBagLayout());

patchesPanel.setBorder(new javax.swing.border.TitledBorder("Patch Update Frequency"));
patchesPanel.setPreferredSize(new java.awt.Dimension(200, 97));
patchFreqAsReleasedRadioButton.setText("As Released");
patchFreqbuttonGroup.add(patchFreqAsReleasedRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqAsReleasedRadioButton, gridBagConstraints);

patchFreqRoutinelyRadioButton.setText("Routinely");
patchFreqbuttonGroup.add(patchFreqRoutinelyRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqRoutinelyRadioButton, gridBagConstraints);

patchFreqAutoRadioButton.setSelected(true);
patchFreqAutoRadioButton.setText("Automatically");
patchFreqbuttonGroup.add(patchFreqAutoRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqAutoRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel1.add(patchesPanel, gridBagConstraints);

avPanel.setLayout(new java.awt.GridBagLayout());

avPanel.setBorder(new javax.swing.border.TitledBorder("Antivirus Update Frequency"));
avPanel.setPreferredSize(new java.awt.Dimension(200, 73));
antiVirusRegRadioButton.setText("Regular");
antiVirusbuttonGroup.add(antiVirusRegRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
avPanel.add(antiVirusRegRadioButton, gridBagConstraints);

antiVirusAutoRadioButton.setSelected(true);
antiVirusAutoRadioButton.setText("Automatically");
```

```
antiVirusbuttonGroup.add(antiVirusAutoRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 10;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
avPanel.add(antiVirusAutoRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel1.add(avPanel, gridBagConstraints);

browserPanel.setLayout(new java.awt.GridBagLayout());

browserPanel.setBorder(new javax.swing.border.TitledBorder("Browser Settings"));
browserPanel.setPreferredSize(new java.awt.Dimension(200, 103));
browserLooseRadioButton.setText("Loose");
browserbuttonGroup.add(browserLooseRadioButton);
browserLooseRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserLooseRadioButton, gridBagConstraints);

browserNormalRadioButton.setText("Normal");
browserbuttonGroup.add(browserNormalRadioButton);
browserNormalRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserNormalRadioButton, gridBagConstraints);

browserStrictRadioButton.setSelected(true);
browserStrictRadioButton.setText("Strict");
browserbuttonGroup.add(browserStrictRadioButton);
browserStrictRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserStrictRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel1.add(browserPanel, gridBagConstraints);

emailPanel.setLayout(new java.awt.GridBagLayout());

emailPanel.setBorder(new javax.swing.border.TitledBorder("Email Settings"));
emailPanel.setPreferredSize(new java.awt.Dimension(200, 97));
emailLooseRadioButton.setText("Loose");
emailbuttonGroup.add(emailLooseRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
emailPanel.add(emailLooseRadioButton, gridBagConstraints);
```

```
emailNormalRadioButton.setText("Normal");
emailbuttonGroup.add(emailNormalRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
emailPanel.add(emailNormalRadioButton, gridBagConstraints);

emailStrictRadioButton.setSelected(true);
emailStrictRadioButton.setText("Strict");
emailbuttonGroup.add(emailStrictRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 18;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
emailPanel.add(emailStrictRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel1.add(emailPanel, gridBagConstraints);

checkboxPanel.setLayout(new java.awt.GridBagLayout());

checkboxPanel.setPreferredSize(new java.awt.Dimension(341, 384));
assetProtectionCheckBox.setText("Access controls fully and correctly confgured at startup");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(assetProtectionCheckBox, gridBagConstraints);

remoteAccessCheckBox.setText("Remote access requires authentication");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(remoteAccessCheckBox, gridBagConstraints);

pKICertsCheckBox.setText("Accept PKI Certs");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(pKICertsCheckBox, gridBagConstraints);

oneTimePassCheckBox.setText("Use one time password token");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(oneTimePassCheckBox, gridBagConstraints);

biometricsCheckBox.setText("Use biometrics");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(biometricsCheckBox, gridBagConstraints);
```

```
tokenPKICheckBox.setText("Use token PKI certs");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(tokenPKICheckBox, gridBagConstraints);

clientPKICheckBox.setText("Use client PKI certs");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(clientPKICheckBox, gridBagConstraints);

vPNClientCheckBox.setText("VPN client");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(vPNClientCheckBox, gridBagConstraints);

scanEmailCheckBox.setText("Scan email attachements");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 10;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(scanEmailCheckBox, gridBagConstraints);

stripEmailCheckBox.setText("Strip email attachments");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(stripEmailCheckBox, gridBagConstraints);

autoLogoutCheckBox.setText("Automatic lock/logout");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(autoLogoutCheckBox, gridBagConstraints);

userAdminNoMACCheckBox.setText("User can administrate non-MAC security attributres");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(userAdminNoMACCheckBox, gridBagConstraints);

userAdminMACCheckBox.setText("MAC security attributes are self-adminstered");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(userAdminMACCheckBox, gridBagConstraints);

adminOnlyInstallsCheckBox.setText("Only administrator can install software");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 15;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
        checkboxPanel.add(adminOnlyInstallsCheckBox, gridBagConstraints);

        blockMediaCheckBox.setText("Block use of removable media and ports");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 16;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(blockMediaCheckBox, gridBagConstraints);

        blockWriteCheckBox.setText("Block write access to local storage");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 17;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(blockWriteCheckBox, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
        boolPanel1.add(checkboxPanel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        catalogComponentPanel.add(boolPanel1, gridBagConstraints);

        catalogComponentScrollPane.setViewportView(catalogComponentPanel);

        add(catalogComponentScrollPane);

    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
    private void softwareRemoveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if(softwareFinalList.getSelectedIndex() >= 0)
        {
            mSoftware_Vector.remove(softwareFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)softwareFinalList.getModel();
            listModel.remove(softwareFinalList.getSelectedIndex());
            softwareFinalList = new JList(listModel);
            softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            softwareFinalScrollPane.setViewportView(softwareFinalList);
        }
    }

    private void softwareMoveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if(softwareSourceList.getSelectedValue() != null)
        {
            Object tempObject = new Object();
            mSoftware_Vector.addElement(softwareSourceList.getSelectedValue().toString());
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < mSoftware_Vector.size(); i++)
            {
                tempObject = mSoftware_Vector.get(i);
                listModel.addElement(tempObject);
            }
            softwareFinalList = null;
            softwareFinalList = new JList(listModel);
```

```
        softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        softwareFinalScrollPane.setViewportView(softwareFinalList);
        software_added = true;
    }
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
      // if the target of the vector is a list box just add the
      // vector directly
public void populateStaticSourceLists(java.util.Vector aBase,
        java.util.Vector aOpSys, java.util.Vector aPassComp,
       java.util.Vector aPassLen, java.util.Vector aSoftware)
{
    //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
    //to populate:
    //base components
    //OS
    //password complexity
    //password length
    //software

    //base component
     hardwareSourceComboBox.removeAllItems();
    for(int i = 0; i < aBase.size(); i++)
    {
        hardwareSourceComboBox.addItem(aBase.get(i));
    }
    //OS
    opSysSourceComboBox.removeAllItems();
    for(int i = 0; i < aOpSys.size(); i++)
    {
        opSysSourceComboBox.addItem(aOpSy s.get(i));
    }

    //software
    softwareSourceList.setListData(aSoftware);
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists(java.util.Vector aSoftware)
{
    softwareSourceList.setListData(aSoftware);
    DefaultListModel listModel = new DefaultListModel();
    Object tempObject = new Object();
    for(int i = 0; i < mSoftware_Vector.size(); i++)
```

```
    {
        tempObject = mSoftware_Vector.get(i);
        listModel.addElement(tempObject);
    }
    softwareFinalList = null;
    softwareFinalList = new JList(listModel);
    softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    softwareFinalScrollPane.setViewportView(softwareFinalList);
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(java.util.Vector aBase,
        java.util.Vector aOpSys, java.util.Vector aPassComp,
       java.util.Vector aPassLen, java.util.Vector aSoftware)
{
    populateStaticSourceLists(aBase,aOpSys, aPassComp, aPassLen, aSoftware);

    softwareMoveButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            softwareMoveButtonActionPerformed(evt);
        }
    });

    softwareRemoveButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            softwareRemoveButtonActionPerformed(evt);
        }
    });
}

public void Load()
{
}

public void Save()
{
}

//the following nametoString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller nametoString() methods are called from the primary
//asset toString() method.
private String softwareList_toString()
{
    String softwareString = new String();
    DefaultListModel listModel = new DefaultListModel();
    if(software_added)
    {
        listModel = (DefaultListModel)softwareFinalList.getModel();
        for(int i = 0; i < listModel.getSize(); i++)
        {
            softwareString +=
            "\tSoftware: " +
            listModel.get(i).toString() +
            " :end\n";
        }
    }
    return softwareString;
}
```

```
//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for a CatalogComponent descriptor
public String toString(String aText)
{
   String outputString = new String();
   outputString =
   "//The start of the component section.\n"+
   "Component:\n\n"+
   "\t//The name of the component.  If \"IsTemplate\" is true, then this"+
   " is a catalogue\n\t//item that the player can purchase and \"Name\""+
   " should be a psuedo\n\t//brand name since it will show up in the"+
   " catalogue.\n\t//Catalogue items can also be referenced in\n"+
   "\t//subsequent Component definitions.  If \"IsTemplate\" is false,"+
   " then this\n\t//component is instantiated at the beginning of the "+
   "game"+
   " and \"Name\" should\n\t//identify the component within the context "+
   "of"+
   " the scenario, e.g.,\n\t//\"Bob's computer.\"\n"+
   "\tName: "+
   nameTextField.getText()+
   " :end\n\n"+
   "\t//If this boolean field is true, this component is a catalogue item"+
   " that is\n\t//part of the buy screen.  If it is false, the component"+
   " is instaniated at the start\n\t//of the game.\n"+
   "\tIsTemplate: "+
   String.valueOf(IsTemplate) +
   " :end\n\n"+
   "\t//A description of the component. If this is a catalogue item, this"+
   " description will\n\t//appear in the ctalogue.  Otherwise, this"+
   " description will be presented to the\n\t//player when the component"+
   " itself is selected and it should be in the context \n\t//the"+
   " scenario.\n"+
   "\tDescription: "+
   descriptionTextArea.getText() +
   " :end\n\n"+
   "\t//The AssetProtection is a boolean value that controls whether the"+
   " intended\n\tassess controls (MAC and DAC) in the asset description"+
   " is fully and\n\t//correctly configured on this machine at startup"+
   ".  If true then it will be correct,\n\t//if false then it will depend"+
   "\n\t//on the skill ratings of the IT department.\n"+
   "\tAssetProtection: "+
   String.valueOf(assetProtectionCheckBox.isSelected()) +
   " :end\n\n"+
   "\t//The base component from which this component is derived.  This"+
   "can\n\t//value can name either another defined \"Component\" from"+
   "within this SDF, or\n\t//it can name one of the pre-defined "+
   "components that are defined outside of the\n\t//SDF.\n"+
   "\tHW: ";
   if(hardwareSourceComboBox.getSelectedItem() != null)
   {
      outputString += hardwareSourceComboBox.getSelectedItem().toString();
   }
   outputString += " :end\n\n"+
   "\t//The dollar cost to purchase the component\n"+
   "\tCost: "+
   costTextField.getText() +
   " :end\n\n"+
   "\t//The dollar value of selling the component after it has been "+
   "purchased\n"+
   "\tResale: "+
```

```
resaleTextField.getText() +
" :end\n\n"+
"\t//The monthly maintenance cost of this component\n"+
"\tMaintenance: "+
maintenanceTextField.getText() +
" :end\n\n"+
"\t//The reliability of the component expressed as a percentage of the"+
" time that is usable\n"+
"\tAvailability: "+
availabilityTextField.getText() +
" :end\n\n"+
"\t//OS (if any) on this component.  Value is from the enumerated list"+
"of possible\n\t//operating systems.  If this value is not present, "+
"the component will have any\n\t//OS defined as part of the component"+
"named in the \"HW\" field.\n"+
"\tOS: ";
if(opSysSourceComboBox.getSelectedItem() != null)
{
   outputString += opSysSourceComboBox.getSelectedItem().toString();
}
outputString += " :end\n\n"+
"\t//List of software programs on this component from the enumerated"+
" list of\n\t//software defined outside of this SDF.  If there are"+
" not \"Software\" fields, then\n\t//the component has whatever "+
"software is defined as part of the component\n\t//named in the"+
"\"HW\" field.  If any fields are present, this component inherits"+
"\n\t//none of the software from its base component.\n"+
softwareList_toString() +
"\n\t//Whether this component requires some form of authenication"+
" when being\n\t//accessed from a remote component.\n"+
"\tRemoteAuthentication: "+
String.valueOf(remoteAccessCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to accept PKI "+
" certificates for\n\t//authentication from remote components."+
"  The assurance depends\n\t//on the OS, this just determines"+
" if the certificates are used.\n"+
"\tAcceptPKICerts: "+
String.valueOf(pKICertsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to accept one-time"+
" passwords\n\t//from password-generation tokens.\n"+
"\tUseOneTimePasswordToken: "+
String.valueOf(oneTimePassCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use biometirc"+
" authentication\n\t//peripherals.\n"+
"\tUseBiometrics: "+
String.valueOf(biometricsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use physical"+
" tokens with\n\t//user-based client PKI certificates for"+
" authentication\n"+
"\tUseTokenPKICerts: "+
String.valueOf(tokenPKICheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use client PKI"+
" certificates managed\n\t//by the client for authenticating to"+
" remote components.  The assurance\n\t//depends on\n\t//the OS"+
", this just determines if the certificates are used.\n"+
"\tUseClientPKICerts: "+
String.valueOf(clientPKICheckBox.isSelected()) +
```

```
" :end\n\n"+
"\t//Whether this component is configured to use local client VPN"+
" software to\n\t//access those remote components that require it"+
" (e.g., a VPN gateway).\n"+
"\tVPNClient: "+
String.valueOf(vPNClientCheckBox.isSelected()) +
" :end\n\n"+
"\t//Email server settings to determine if incoming e-mail"+
" attachments\n\t//are scanned for viruses\n\t//and/or are stripped.\n"+
"\tScanEmailAttachments: "+
String.valueOf(scanEmailCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether to strip email attachments,\n"+
"\tStripEmailAttachments: "+
String.valueOf(stripEmailCheckBox.isSelected()) +
" :end\n\n"+
"\t//Automatic lock/logout, determines if the workstation"+
" automatically requires\n\t//a password to restart activity after"+
" a timeout of the specified duration.\n\t//Boolean\n"+
"\tAutomaticLockLogout: "+
String.valueOf(autoLogoutCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether the non-MAC security attributes of this component are "+
"self\n\t//administered by the user who has access to the component."+
"\n\t//If \"true\", the other component configuration values are"+
" liable to \n\t//change based on user training, competence and"+
" trustworthiness.\n"+
"\tSelfAdminister: "+
String.valueOf(userAdminNoMACCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether the MAC security attributes of this component are"+
"self-administered.\n"+
"\tSelfAdministerMAC: "+
String.valueOf(userAdminMACCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether software can only be installed by administrator.\n"+
"\tAdministerSoftwareControl: "+
String.valueOf(adminOnlyInstallsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Block use of removable media and ports (e.g., USB)"+
"\n\t//(Affects ablility of malicious insider to extract sensitive"+
" data.  Also affects\n\t//chance of user introducing malicious"+
" software.)\n"+
"\tBlockRemovableMedia: "+
String.valueOf(blockMediaCheckBox.isSelected()) +
" :end\n\n"+
"\t//Block write access to local storage (Affects cost of theft of"+
" workstation, e.g.,\n\t//if the value is \"true\", then no assets"+
" are stored locally.\n"+
"\tBlockLocalStorage: "+
String.valueOf(blockWriteCheckBox.isSelected()) +
" :end\n\n"+
"\t//Browser settings. The most strict selected will be enacted\n"+
"\tBrowserSettingsLoose: "+
String.valueOf(browserLooseRadioButton.isSelected()) +
" :end\n"+
"\tBrowserSettingsNormal: "+
String.valueOf(browserNormalRadioButton.isSelected()) +
" :end\n"+
"\tBrowserSettingsStrict: "+
String.valueOf(browserStrictRadioButton.isSelected()) +
" :end\n\n"+
```

```
    "\t//Email settings. The most strict will be enacted\n"+
    "\tEmailSettingsLoose: "+
    String.valueOf(emailLooseRadioButton.isSelected())+
    " :end\n"+
    "\tEmailSettingsNormal: "+
    String.valueOf(emailNormalRadioButton.isSelected()) +
    " :end\n"+
    "\tEmailSettingsStrict: "+
    String.valueOf(emailStrictRadioButton.isSelected()) +
    " :end\n\n"+
    "\t//Frequency at which patches are applied. The most strict will be" +
    " enacted\n"+
    "\tUpdatePatchesAsReleased: "+
    String.valueOf(patchFreqAsReleasedRadioButton.isSelected())+
    " :end\n"+
    "\tUpdatePatchesRoutinely: "+
    String.valueOf(patchFreqRoutinelyRadioButton.isSelected()) +
    " :end\n"+
    "\tUpdatePatchesAutomatically: "+
    String.valueOf(patchFreqAutoRadioButton.isSelected()) +
    " :end\n\n"+
    "\t//Frequency of antivirus updates.\n" +
    "\tUpdateAntivirusRegular: "+
    String.valueOf(antiVirusRegRadioButton.isSelected()) +
    " :end\n"+
    "\tUpdateAntivirusAutomatic: "+
    String.valueOf(antiVirusAutoRadioButton.isSelected()) +
    " :end\n\n"+
    ":end//Of The Component Section\n\n";
    return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
    return nameTextField.getText();
}

// Variables declaration - do not modify
private javax.swing.JCheckBox adminOnlyInstallsCheckBox;
private javax.swing.JRadioButton antiVirusAutoRadioButton;
private javax.swing.JRadioButton antiVirusRegRadioButton;
private javax.swing.ButtonGroup antiVirusbuttonGroup;
private javax.swing.JCheckBox assetProtectionCheckBox;
private javax.swing.JCheckBox autoLogoutCheckBox;
private javax.swing.JPanel avPanel;
private javax.swing.JTextField availabilityTextField;
private javax.swing.JLabel availablityLabel;
private javax.swing.JCheckBox biometricsCheckBox;
private javax.swing.JCheckBox blockMediaCheckBox;
private javax.swing.JCheckBox blockWriteCheckBox;
private javax.swing.JPanel boolPanel1;
private javax.swing.JRadioButton browserLooseRadioButton;
private javax.swing.JRadioButton browserNormalRadioButton;
private javax.swing.JPanel browserPanel;
private javax.swing.JRadioButton browserStrictRadioButton;
private javax.swing.ButtonGroup browserbuttonGroup;
private javax.swing.JPanel catalogComponentPanel;
private javax.swing.JScrollPane catalogComponentScrollPane;
private javax.swing.JPanel checkboxPanel;
private javax.swing.JCheckBox clientPKICheckBox;
```

```
    private javax.swing.JPanel comboBoxandTextPanel1;
    private javax.swing.JLabel costLabel;
    private javax.swing.JTextField costTextField;
    private javax.swing.JLabel descriptionLabel;
    private javax.swing.JScrollPane descriptionScrollPane;
    private javax.swing.JTextArea descriptionTextArea;
    private javax.swing.JRadioButton emailLooseRadioButton;
    private javax.swing.JRadioButton emailNormalRadioButton;
    private javax.swing.JPanel emailPanel;
    private javax.swing.JRadioButton emailStrictRadioButton;
    private javax.swing.ButtonGroup emailbuttonGroup;
    private javax.swing.JLabel hardwareNameLabel;
    private javax.swing.JComboBox hardwareSourceComboBox;
    private javax.swing.JPanel listsPanel;
    private javax.swing.JLabel maintenanceLabel;
    private javax.swing.JTextField maintenanceTextField;
    private javax.swing.JPanel manyToOnePanel;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JCheckBox oneTimePassCheckBox;
    private javax.swing.JLabel opSysLabel;
    private javax.swing.JComboBox opSysSourceComboBox;
    private javax.swing.JCheckBox pKICertsCheckBox;
    private javax.swing.JRadioButton patchFreqAsReleasedRadioButton;
    private javax.swing.JRadioButton patchFreqAutoRadioButton;
    private javax.swing.JRadioButton patchFreqRoutinelyRadioButton;
    private javax.swing.ButtonGroup patchFreqbuttonGroup;
    private javax.swing.JPanel patchesPanel;
    private javax.swing.JCheckBox remoteAccessCheckBox;
    private javax.swing.JLabel resaleLabel;
    private javax.swing.JTextField resaleTextField;
    private javax.swing.JCheckBox scanEmailCheckBox;
    private javax.swing.JList softwareFinalList;
    private javax.swing.JScrollPane softwareFinalScrollPane;
    private javax.swing.JLabel softwareLabel;
    private javax.swing.JButton softwareMoveButton;
    private javax.swing.JButton softwareRemoveButton;
    private javax.swing.JList softwareSourceList;
    private javax.swing.JScrollPane softwareSourceScrollPane;
    private javax.swing.JCheckBox stripEmailCheckBox;
    private javax.swing.JCheckBox tokenPKICheckBox;
    private javax.swing.JCheckBox userAdminMACCheckBox;
    private javax.swing.JCheckBox userAdminNoMACCheckBox;
    private javax.swing.JCheckBox vPNClientCheckBox;
    // End of variables declaration
    private boolean IsTemplate;
    private boolean software_added;
    private Vector mSoftware_Vector;
}
```

F. THE SOURCE OF: CONDITION

```
/*
 * Condition.java
 *
 * Created on January 23, 2004, 9:50 AM
 */

package CCSDT;

/**
 *
```

```
 * @author  KJohns
 *This class is the graphical representation of a Condition descriptor.
 *This is where the Condition form is managed and the Condition
 *toString() resides.
 */
public class Condition extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Condition */
    public Condition()
    {
        initComponents();
        conditionINIVec = new java.util.Vector();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        conditionScrollPane = new javax.swing.JScrollPane();
        conditionPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        conditionClassLabel = new javax.swing.JLabel();
        conditionClassComboBox = new javax.swing.JComboBox();
        conditionClassTextField = new javax.swing.JTextField();
        parameter1Label = new javax.swing.JLabel();
        parameter1TextField = new javax.swing.JTextField();
        parameter2Label = new javax.swing.JLabel();
        parameter2TextField = new javax.swing.JTextField();
        parameter3Label = new javax.swing.JLabel();
        parameter3TextField = new javax.swing.JTextField();
        parameter4Label = new javax.swing.JLabel();
        parameter4TextField = new javax.swing.JTextField();
        parameter5Label = new javax.swing.JLabel();
        parameter5TextField = new javax.swing.JTextField();

        setLayout(new java.awt.GridLayout(1, 0));

        conditionPanel.setLayout(new java.awt.GridBagLayout());

        conditionPanel.setPreferredSize(new java.awt.Dimension(397, 169));
        nameLabel.setLabelFor(nameTextField);
        nameLabel.setText("Condition Name:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        conditionPanel.add(nameLabel, gridBagConstraints);

        nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        conditionPanel.add(nameTextField, gridBagConstraints);

        conditionClassLabel.setText("Condition Class:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        conditionPanel.add(conditionClassLabel, gridBagConstraints);

        conditionClassComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
        conditionClassComboBox.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        conditionClassComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
conditionPanel.add(conditionClassComboBox, gridBagConstraints);

conditionClassTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
conditionPanel.add(conditionClassTextField, gridBagConstraints);

parameter1Label.setText("Parameter 1:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
conditionPanel.add(parameter1Label, gridBagConstraints);

parameter1TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
conditionPanel.add(parameter1TextField, gridBagConstraints);

parameter2Label.setText("Parameter 2:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
conditionPanel.add(parameter2Label, gridBagConstraints);

parameter2TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
conditionPanel.add(parameter2TextField, gridBagConstraints);

parameter3Label.setText("Parameter 3:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
conditionPanel.add(parameter3Label, gridBagConstraints);

parameter3TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
conditionPanel.add(parameter3TextField, gridBagConstraints);

parameter4Label.setText("Parameter 4:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
```

```
    gridBagConstraints.gridy = 5;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    conditionPanel.add(parameter4Label, gridBagConstraints);

    parameter4TextField.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 5;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    conditionPanel.add(parameter4TextField, gridBagConstraints);

    parameter5Label.setText("Parameter 5:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 6;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    conditionPanel.add(parameter5Label, gridBagConstraints);

    parameter5TextField.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 6;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    conditionPanel.add(parameter5TextField, gridBagConstraints);

    conditionScrollPane.setViewportView(conditionPanel);

    add(conditionScrollPane);

}

private void conditionClassComboBoxActionPerformed(java.awt.event.ActionEvent evt)
{
    if(conditionClassComboBox.getSelectedItem() != null)
    {
        conditionClassTextField.setText(
                    conditionClassComboBox.getSelectedItem().toString());
        Object tempObj = new Object();
        java.util.Vector tempVec = new java.util.Vector();
        tempObj = conditionINIVec.get(
                        conditionClassComboBox.getSelectedIndex());
        tempVec = (java.util.Vector)tempObj;
        for(int i = 0; i < tempVec.size(); i++)
        {
            switch(i)
            {
                case 1:
                    parameter1Label.setText(tempVec.get(1).toString()+":");
                    parameter2Label.setText("Parameter 2:");
                    parameter3Label.setText("Parameter 3:");
                    parameter4Label.setText("Parameter 4:");
                    parameter5Label.setText("Parameter 5:");
                    break;
                case 2:
                    parameter2Label.setText(tempVec.get(2).toString()+":");
                    parameter3Label.setText("Parameter 3:");
                    parameter4Label.setText("Parameter 4:");
                    parameter5Label.setText("Parameter 5:");
                    break;
                case 3:
                    parameter3Label.setText(tempVec.get(3).toString()+":");
                    parameter4Label.setText("Parameter 4:");
```

```
                parameter5Label.setText("Parameter 5:");
                break;
            case 4:
                parameter4Label.setText(tempVec.get(4).toString()+":");
                parameter5Label.setText("Parameter 5:");
                break;
            case 5:
                parameter5Label.setText(tempVec.get(5).toString()+":");
                break;
            default:
                parameter1Label.setText("Parameter 1:");
                parameter2Label.setText("Parameter 2:");
                parameter3Label.setText("Parameter 3:");
                parameter4Label.setText("Parameter 4:");
                parameter5Label.setText("Parameter 5:");
                break;
        }
      }
   }
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
     // if the target of the vector is a list box just add the
     // vector directly
public void populateStaticSourceLists(java.util.Vector aCondition)
{
   //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
   //to populate:
   //conditions

   conditionINIVec = aCondition;
   //conditions
   conditionClassComboBox.removeAllItems();
   Object tempObj = new Object();
   java.util.Vector tempVec = new java.util.Vector();
   for(int i = 0; i < aCondition.size(); i++)
   {
      tempObj = aCondition.get(i);
      tempVec = (java.util.Vector)tempObj;
      conditionClassComboBox.addItem(tempVec.get(0));
   }
}

public void Load()
{
}

public void Save()
```

```
{
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for a Condition descriptor
public String toString(String aText)
{
   String outputString = new String();
   outputString +=
   "\tCondition:\n"+
   "\t\t//Enumeration value that defines the condition class of this condition.\n\n"+
   "\t\t//Example condition classes include:  MinNetValueOfEnterprise,\n"+
   "\t\t//MinUserMorale, NumberSuccessfulSecurityAttacks, etc.\n"+
   "\t\tConditionClass: "+conditionClassTextField.getText()+" :end\n\n"+
   "\t\t//Tagnames are 1 word only and are used to bind conditions to triggers\n"+
   "\t\tTagname: "+nameTextField.getText()+" :end\n\n"+
   "\t\t//The condition class determines the number of parameters and their\n"+
   "\t\t//type (integer, string, boolean, or enumeration)\n\n"+
   "\t\t//The number and type of parameters is fixed for each condition class.\n";
   if(parameter1TextField.getText().length() > 0)
   {
      outputString += "\n\t\tParameter: "+parameter1TextField.getText()+" :end";
   }
   if(parameter2TextField.getText().length() > 0)
   {
      outputString += "\n\t\tParameter: "+parameter2TextField.getText()+" :end";
   }
   if(parameter3TextField.getText().length() > 0)
   {
      outputString += "\n\t\tParameter: "+parameter3TextField.getText()+" :end";
   }
   if(parameter4TextField.getText().length() > 0)
   {
      outputString += "\n\t\tParameter: "+parameter4TextField.getText()+" :end";
   }
   if(parameter5TextField.getText().length() > 0)
   {
      outputString += "\n\t\tParameter: "+parameter5TextField.getText()+" :end";
   }
   outputString += "\n\n\t:end\n\n";

   return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
   return nameTextField.getText();
}

//This method is used in trigger to populate the condition list text area
//The method takes the data in condition and formats the output for the
//text area
public String getTriggerString()
{
   String triggerString = new String();
   triggerString += "(NAME)"+nameTextField.getText() + "\t" +
   "(CONDITON CLASS)"+conditionClassTextField.getText() + "\t";
   if(parameter1TextField != null)
```

```
        {
            triggerString += "(PARAMETER #1)"+parameter1TextField.getText() + "\t";
        }
        if(parameter2TextField != null)
        {
            triggerString += "(PARAMETER #2)"+parameter2TextField.getText() + "\t";
        }
        if(parameter3TextField != null)
        {
            triggerString += "(PARAMETER #3)"+parameter3TextField.getText() + "\t";
        }
        if(parameter4TextField != null)
        {
            triggerString += "(PARAMETER #4)"+parameter4TextField.getText() + "\t";
        }
        if(parameter5TextField != null)
        {
            triggerString += "(PARAMETER #5)"+parameter5TextField.getText();
        }
        triggerString += "\n";
        return triggerString;
    }

    //This method provides a way to reinitialize any miscellaneous
    //(non-list, non-button) listeners when they die after IO operations.
    //The code is taken line for line from  the initComponents() method.
    public void reInitListeners(java.util.Vector aCondition)
    {
        populateStaticSourceLists(aCondition);
        conditionClassComboBox.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                conditionClassComboBoxActionPerformed(evt);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JComboBox conditionClassComboBox;
    private javax.swing.JLabel conditionClassLabel;
    private javax.swing.JTextField conditionClassTextField;
    private javax.swing.JPanel conditionPanel;
    private javax.swing.JScrollPane conditionScrollPane;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameText Field;
    private javax.swing.JLabel parameter1Label;
    private javax.swing.JTextField parameter1TextField;
    private javax.swing.JLabel parameter2Label;
    private javax.swing.JTextField parameter2TextField;
    private javax.swing.JLabel parameter3Label;
    private javax.swing.JTextField parameter3TextField;
    private javax.swing.JLabel parameter4Label;
    private javax.swing.JTextField parameter4TextField;
    private javax.swing.JLabel parameter5Label;
    private javax.swing.JTextField parameter5TextField;
    // End of variables declaration
    private java.util.Vector conditionINIVec;
}
```

G. THE SOURCE OF: COST LIST

```
/*
 * CostList.java
 *
 * Created on February 2, 2004, 2:05 PM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is used as an intermediate container to store user choices for
 *a cost list.
 */
public class CostList implements java.io.Serializable
{

    /** Creates a new instance of CostList */
    public CostList()
    {
        user = new String();
        group = new String();
        read = new String();
        write = new String();
        control = new String();
        execute = new String();
        cost = new String();
        costChange = new String();
        attackerMotive = new String();
        attackerValueChange = new String();
    }

    public void setUser(String aUser)
    {
        user = aUser;
    }
    public void setGroup(String aGroup)
    {
        group = aGroup;
    }
    public void setRead(String aRead)
    {
        read = aRead;
    }
    public void setWrite(String aWrite)
    {
        write = aWrite;
    }
    public void setControl(String aControl)
    {
        control = aControl;
    }
    public void setExecute(String aExecute)
    {
        execute = aExecute;
    }
    public void setCost(String aCost)
    {
        cost = aCost;
    }
    public void setCostChange(String aCostChange)
    {
```

```
        costChange = aCostChange;
    }
    public void setAttackerMotive(String aAttackerMotive)
    {
        attackerMotive = aAttackerMotive;
    }
    public void setAttackerValueChange(String aAttackerValueChange)
    {
        attackerValueChange = aAttackerValueChange;
    }
    public String getUser()
    {
        return user;
    }
    public String getGroup()
    {
        return group;
    }
    public String getRead()
    {
        return read;
    }
    public String getWrite()
    {
        return write;
    }
    public String getControl()
    {
        return control;
    }
    public String getExecute()
    {
        return execute;
    }
    public String getCost()
    {
        return cost;
    }
    public String getCostChange()
    {
        return costChange;
    }
    public String getAttackerMotive()
    {
        return attackerMotive;
    }
    public String getAttackerValueChange()
    {
        return attackerValueChange;
    }
    public String getCostList()
    {
        return "(USER.GROUP)"+user + "." + group + "(READ)" + read + "(WRITE)" + write +
"(CONTROL)" +
            control + "(EXECUTE)" + execute + "(COST)" + cost + "(COST CHANGE)" + costChange +
            "(ATTACKER MOTIVE)" + attackerMotive + "(ATTACKER VALUE CHANGE)" +
attackerValueChange;
    }

    private String user;
    private String group;
    private String read;
```

```
    private String write;
    private String control;
    private String execute;
    private String cost;
    private String costChange;
    private String attackerMotive;
    private String attackerValueChange;

}
```

H. THE SOURCE OF: DAC GROUP

```
/*
 * DACGroup.java
 *
 * Created on January 23, 2004, 9:54 AM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is the graphical representation of a DACGroup descriptor.
 *This is where the DACGroup form is managed and the DACGroup
 *toString() resides.
 */
public class DACGroup extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form DACGroup */
    public DACGroup()
    {
        initComponents();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        dacGroupScrollPane = new javax.swing.JScrollPane();
        dacGroupPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        initialBackgroundCheckLabel = new javax.swing.JLabel();
        initialBackgroundCheckComboBox = new javax.swing.JComboBox();

        setLayout(new java.awt.GridLayout(1, 0));

        dacGroupPanel.setLayout(new java.awt.GridBagLayout());

        nameLabel.setLabelFor(nameTextField);
        nameLabel.setText("DAC Group Name:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        dacGroupPanel.add(nameLabel, gridBagConstraints);

        nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        dacGroupPanel.add(nameTextField, gridBagConstraints);

        initialBackgroundCheckLabel.setLabelFor(initialBackgroundCheckComboBox);
```

```
    initialBackgroundCheckLabel.setText("Initial Background Check:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    dacGroupPanel.add(initialBackgroundCheckLabel, gridBagConstraints);

    initialBackgroundCheckComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    dacGroupPanel.add(initialBackgroundCheckComboBox, gridBagConstraints);

    dacGroupScrollPane.setViewportView(dacGroupPanel);

    add(dacGroupScrollPane);

}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
      // if the target of the vector is a list box just add the
      // vector directly
public void populateStaticSourceLists(java.util.Vector aBackgroundcheck)
{
    //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
    //to populate:
    //background check

    //base component
    for(int i = 0; i < aBackgroundcheck.size(); i++)
    {
        initialBackgroundCheckComboBox.addItem(aBackgroundcheck.get(i));
    }
}

public void Load()
{
}

public void Save()
{
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an DACGroup descriptor
```

```
    public String toString(String aText)
    {
        String outputString = new String();
        outputString +=
        "\t//Group and background check level\n"+
        "\tGroup: "+nameTextField.getText()+" :end\n"+
        "\tInitialBackGroundCheck: "+
        initialBackgroundCheckComboBox.getSelectedItem().toString()+
        " :end    //High, Med, Low, None\n\n";
        return outputString;
    }

    //since the name field of the form is private this method provides a way
    //to fetch the name.
    public String getNameTextField()
    {
        return nameTextField.getText();
    }

    //This method provides a way to reinitialize any miscellaneous
    //(non-list, non-button) listeners when they die after IO operations.
    //The code is taken line for line from  the initComponents() method.
    public void reInitListeners(java.util.Vector aBackgroundcheck)
    {
        populateStaticSourceLists(aBackgroundcheck);
    }

    // Variables declaration - do not modify
    private javax.swing.JPanel dacGroupPanel;
    private javax.swing.JScrollPane dacGroupScrollPane;
    private javax.swing.JComboBox initialBackgroundCheckComboBox;
    private javax.swing.JLabel initialBackgroundCheckLabel;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    // End of variables declaration

}
```

I. THE SOURCE OF: DEPARTMENT

```
/*
 * Department.java
 *
 * Created on January 23, 2004, 10:01 AM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is the graphical representation of a Department descriptor.
 *This is where the Department form is managed and the Department
 *toString() resides.
 */
public class Department extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Department */
    public Department()
    {
        initComponents();
```

```
}

private void initComponents() {
   java.awt.GridBagConstraints gridBagConstraints;

   departmentScrollPane = new javax.swing.JScrollPane();
   departmentPanel = new javax.swing.JPanel();
   nameLabel = new javax.swing.JLabel();
   nameTextField = new javax.swing.JTextField();

   setLayout(new java.awt.GridLayout(1, 0));

   departmentPanel.setLayout(new java.awt.GridBagLayout());

   departmentPanel.setPreferredSize(new java.awt.Dimension(471, 119));
   nameLabel.setLabelFor(nameTextField);
   nameLabel.setText("Department Name:");
   gridBagConstraints = new java.awt.GridBagConstraints();
   gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
   departmentPanel.add(nameLabel, gridBagConstraints);

   nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
   gridBagConstraints = new java.awt.GridBagConstraints();
   gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
   departmentPanel.add(nameTextField, gridBagConstraints);

   departmentScrollPane.setViewportView(departmentPanel);

   add(departmentScrollPane);

}

public void Load()
{
}

public void Save()
{
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an Department descriptor
public String toString(String aText)
{
   String outputString = new String();
   outputString +=
   "//The name of a department\n"+
   "Department:\n"+
   "\tName: "+ nameTextField.getText()+" :end\n"+
   ":end     //of Department\n\n";
   return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
   return nameTextField.getText();
}
```

```
    // Variables declaration - do not modify
    private javax.swing.JPanel departmentPanel;
    private javax.swing.JScrollPane departmentScrollPane;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JText Field nameTextField;
    // End of variables declaration

}
```

J. THE SOURCE OF: FILE NODE

```
/*
 * FileNode.java
 *
 * Created on November 21, 2003, 2:23 PM
 */

package CCSDT;

import javax.swing.tree.*;
import java.io.File;
/**
 *
 * @author  kjohns
 *This class is adapted from code found in:
 *Horstmann, C & Cornell, G. (2002). Core Java 2, Volume II: Advanced Features.
 *Palo Alto: Sun Microsystems Press.
 *
 *This class takes a file object and recursively explores its file path to
 *populate a tree
 */
    public class FileNode extends DefaultMutableTreeNode {
            private boolean explored = false;

            public FileNode(File file)
        {
                        setUserObject(file);
            }

            public boolean getAllowsChildren()
        {
          return isDirectory();
        }

            public boolean isLeaf()
        {
          return !isDirectory();
        }

            public File getFile()
        {
          return (File)getUserObject();
        }
            public void explore()
        {
          explore(false);
        }

            public boolean isExplored()
        {
          return explored;
        }
```

```
        public boolean isDirectory()
    {
                File file = (File)getUserObject();
                return file.isDirectory();
        }
        public String toString()
    {
                File file = (File)getUserObject();
                String filename = file.toString();
                int index = filename.lastIndexOf("\\");

                return (index != -1 && index != filename.length()-1) ?
                                                        filename.substring(index+1) :
                                                        filename;
        }

        public void explore(boolean force)
    {
                if(!isExplored() || force)
        {
                        File file = getFile();
                        File[] children = file.listFiles();

                        for(int i=0; i < children.length; ++i)
                                add(new FileNode(children[i]));

                        explored = true;
                }
        }
  }//end of FileNode
```

K. THE SOURCE OF: FILTER

```
/*
 * Filter.java
 *
 * Created on January 23, 2004, 10:19 AM
 */

package CCSDT;
import java.util.*;
import javax.swing.*;
import java.io.*;

/**
 *
 * @author  KJohns
 *This class is the graphical representation of a Filter descriptor.
 *This is where the Filter form is managed and the Filter
 *toString() resides.
 */
public class Filter extends ScenarioElement implements java.io.Serializable
{

   /** Creates new form Filter */
   public Filter()
   {
      initComponents();
      mApplicationVector = new Vector();
      mComponentVector = new Vector();
      mNetworkVector = new Vector();
```

```
    application_added = false;
    component_added = false;
    network_added = false;
}

private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;

    filterScrollPane = new javax.swing.JScrollPane();
    filterPanel = new javax.swing.JPanel();
    textAndComboPanel = new javax.swing.JPanel();
    nameLabel = new javax.swing.JLabel();
    nameTextField = new javax.swing.JTextField();
    directionLabel = new javax.swing.JLabel();
    directionComboBox = new javax.swing.JComboBox();
    networkAppliedLabel = new javax.swingJLabel();
    networkAppliedComboBox = new javax.swing.JComboBox();
    networkAppliedFinalTextField = new javax.swing.JTextField();
    listsPanel = new javax.swing.JPanel();
    applicationLabel = new javax.swing.JLabel();
    applicationSourceScrollPane = new javax.swing.JScrollPane();
    applicationSourceList = new javax.swing.JList();
    applicationMoveButton = new javax.swing.JButton();
    applicationFinalScrollPane = new javax.swing.JScrollPane();
    applicationFinalList = new javax.swing.JList();
    applicationRemoveButton = new javax.swing.JButton();
    componentLabel = new javax.swing.JLabel();
    componentSourceScrollPane = new javax.swing.JScrollPane();
    componentSourceList = new javax.swing.JList();
    componentMoveButton = new javax.swing.JButton();
    componentFinalScrollPane = new javax.swing.JScrollPane();
    componentFinalList = new javax.swing.JList();
    componentRemoveButton = new javax.swing.JButton();
    networkLabel = new javax.swing.JLabel();
    networkSourceScrollPane = new javax.swing.JScrollPane();
    networkSourceList = new javax.swing.JList();
    networkMoveButton = new javax.swing.JButton();
    networkFinalScrollPane = new javax.swing.JScrollPane();
    networkFinalList = new javax.swing.JList();
    networkRemoveButton = new javax.swing.JButton();
    boolPanel = new javax.swing.JPanel();
    blockAllCompanyExternalCheckBox = new javax.swing.JCheckBox();
    blockAllSiteExternalCheckBox = new javax.swing.JCheckBox();
    blockAllNetworkExternalCheckBox = new javax.swing.JCheckBox();

    setLayout(new java.awt.GridLayout(1, 0));

    filterPanel.setLayout(new java.awt.GridBagLayout());

    filterPanel.setPreferredSize(new java.awt.Dimension(692, 715));
    textAndComboPanel.setLayout(new java.awt.GridBagLayout());

    nameLabel.setLabelFor(nameTextField);
    nameLabel.setText("Filter Name:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    gridBagConstraints.insets = new java.awt.Insets(0, 35, 0, 0);
    textAndComboPanel.add(nameLabel, gridBagConstraints);

    nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
textAndComboPanel.add(nameTextField, gridBagConstraints);

directionLabel.setLabelFor(directionComboBox);
directionLabel.setText("Filter Blocking Direction:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(directionLabel, gridBagConstraints);

directionComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
textAndComboPanel.add(directionComboBox, gridBagConstraints);

networkAppliedLabel.setText("Network Filter is Applied To:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(networkAppliedLabel, gridBagConstraints);

networkAppliedComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
networkAppliedComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      networkAppliedComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(networkAppliedComboBox, gridBagConstraints);

networkAppliedFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(networkAppliedFinalTextField, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
filterPanel.add(textAndComboPanel, gridBagConstraints);

listsPanel.setLayout(new java.awt.GridBagLayout());

applicationLabel.setLabelFor(applicationSourceList);
applicationLabel.setText("Applications Filtered:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(applicationLabel, gridBagConstraints);

applicationSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
applicationSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
applicationSourceScrollPane.setViewportView(applicationSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(applicationSourceScrollPane, gridBagConstraints);

applicationMoveButton.setText(">>>>>");
applicationMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      applicationMoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
listsPanel.add(applicationMoveButton, gridBagConstraints);

applicationFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
applicationFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
applicationFinalScrollPane.setViewportView(applicationFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
listsPanel.add(applicationFinalScrollPane, gridBagConstraints);

applicationRemoveButton.setText("Remove");
applicationRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      applicationRemoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 35, 0, 0);
listsPanel.add(applicationRemoveButton, gridBagConstraints);

componentLabel.setLabelFor(componentSourceList);
componentLabel.setText("Components Protected by this Filter:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(componentLabel, gridBagConstraints);

componentSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
componentSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
componentSourceScrollPane.setViewportView(componentSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(componentSourceScrollPane, gridBagConstraints);

componentMoveButton.setText(">>>>>");
componentMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        componentMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 5;
listsPanel.add(componentMoveButton, gridBagConstraints);

componentFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
componentFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
componentFinalScrollPane.setViewportView(componentFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
listsPanel.add(componentFinalScrollPane, gridBagConstraints);

componentRemoveButton.setText("Remove");
componentRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        componentRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 35, 0, 0);
listsPanel.add(componentRemoveButton, gridBagConstraints);

networkLabel.setLabelFor(networkSourceList);
networkLabel.setText("Networks Protected by this Filter:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(networkLabel, gridBagConstraints);

networkSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
networkSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
networkSourceScrollPane.setViewportView(networkSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(networkSourceScrollPane, gridBagConstraints);

networkMoveButton.setText(">>>>>");
networkMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        networkMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 7;
listsPanel.add(networkMoveButton, gridBagConstraints);
```

```
                networkFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
                networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
                networkFinalScrollPane.setViewportView(networkFinalList);

                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 3;
                gridBagConstraints.gridy = 7;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                listsPanel.add(networkFinalScrollPane, gridBagConstraints);

                networkRemoveButton.setText("Remove");
                networkRemoveButton.addActionListener(new java.awt.event.ActionListener() {
                   public void actionPerformed(java.awt.event.ActionEvent evt) {
                      networkRemoveButtonActionPerformed(evt);
                   }
                });

                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 3;
                gridBagConstraints.gridy = 8;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                gridBagConstraints.insets = new java.awt.Insets(0, 35, 0, 0);
                listsPanel.add(networkRemoveButton, gridBagConstraints);

                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 1;
                gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
                filterPanel.add(listsPanel, gridBagConstraints);

                boolPanel.setLayout(new java.awt.GridBagLayout());

                blockAllCompanyExternalCheckBox.setText("Block all traffic to/from components outside of the
company");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 9;
                gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(blockAllCompanyExternalCheckBox, gridBagConstraints);

                blockAllSiteExternalCheckBox.setText("Block all traffic to/from components not located at the same
site");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 10;
                gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(blockAllSiteExternalCheckBox, gridBagConstraints);

                blockAllNetworkExternalCheckBox.setText("Block all traffic to/from components not located on the
same inbound network");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 11;
                gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(blockAllNetworkExternalCheckBox, gridBagConstraints);

                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
```

```
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        filterPanel.add(boolPanel, gridBagConstraints);

        filterScrollPane.setViewportView(filterPanel);

        add(filterScrollPane);

    }
//the following listener sets a text field based on a combo box selection
    private void networkAppliedComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(networkAppliedComboBox.getSelectedItem() != null)
        {
            networkAppliedFinalTextField.setText(
                    networkAppliedComboBox.getSelectedItem().toString());
        }
    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
    private void networkRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(networkFinalList.getSelectedIndex() >= 0)
        {
            mNetworkVector.remove(networkFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)networkFinalList.getModel();
            listModel.remove(networkFinalList.getSelectedIndex());
            networkFinalList = new JList(listModel);
            networkFinalList.setSelectionMode(
                        javax.swing.ListSelectionModel.SINGLE_SELECTION);
            networkFinalScrollPane.setViewportView(networkFinalList);
        }
    }

    private void networkMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(networkSourceList.getSelectedValue() != null)
        {
            Object tempObject = new Object();
            mNetworkVector.addElement(networkSourceList.getSelectedValue().toString());
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < mNetworkVector.size(); i++)
            {
                tempObject = mNetworkVector.get(i);
                listModel.addElement(tempObject);
            }
            networkFinalList = null;
            networkFinalList = new JList(listModel);
            networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            networkFinalScrollPane.setViewportView(networkFinalList);
            network_added = true;
        }
    }

    private void componentRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(componentFinalList.getSelectedIndex() >= 0)
        {
            mComponentVector.remove(componentFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)componentFinalList.getModel();
            listModel.remove(componentFinalList.getSelectedIndex());
```

```
        componentFinalList = new JList(listModel);
        componentFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        componentFinalScrollPane.setViewportView(componentFinalList);
    }
}

private void componentMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(componentSourceList.getSelectedValue() != null)
    {
        Object tempObject = new Object();
        mComponentVector.addElement(componentSourceList.getSelectedValue().toString());
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < mComponentVector.size(); i++)
        {
            tempObject = mComponentVector.get(i);
            listModel.addElement(tempObject);
        }
        componentFinalList = null;
        componentFinalList = new JList(listModel);
        componentFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        componentFinalScrollPane.setViewportView(componentFinalList);
        component_added = true;
    }
}

private void applicationRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(applicationFinalList.getSelectedIndex() >= 0)
    {
        mApplicationVector.remove(applicationFinalList.getSelectedIndex());
        DefaultListModel listModel = new DefaultListModel();
        listModel = (DefaultListModel)applicationFinalList.getModel();
        listModel.remove(applicationFinalList.getSelectedIndex());
        applicationFinalList = new JList(listModel);
        applicationFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        applicationFinalScrollPane.setViewportView(applicationFinalList);
    }
}

private void applicationMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(applicationSourceList.getSelectedValue() != null)
    {
        Object tempObject = new Object();
        mApplicationVector.addElement(applicationSourceList.getSelectedValue().toString());
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < mApplicationVector.size(); i++)
        {
            tempObject = mApplicationVector.get(i);
            listModel.addElement(tempObject);
        }
        applicationFinalList = null;
        applicationFinalList = new JList(listModel);
        applicationFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        applicationFinalScrollPane.setViewportView(applicationFinalList);
        application_added = true;
    }
}

public void Load()
{
```

```
}

public void Save()
{
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateDynamicSourceLists() takes datapath and startupScenario as
//parameters in ScenarioDefinitionTool.java and is used to add dynamic
//content to combo boxes and list boxes.
//First the scenairo is used to get the added reusable sets
//Second any combo boxes to be used are cleared and reinitialized
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
{
   //uses the Scenario passed to it to set the dynamic values of
   //dependent sets
   //Network
   //PhysicalComponent
   Object tempObj = new Object();
   Vector tempVec = new Vector();
   ScenarioElementSet tempSet = new ScenarioElementSet();
   physicalComponentListVec = new Vector();
   networkListVec = new Vector();
   componentSourceList.removeAll();
   networkSourceList.removeAll();
   //physical component
   tempObj = aScenario.scenarioManager.get(8);
   tempVec = (Vector)tempObj;
   for(int i = 0; i < tempVec.size(); i++)
   {
      tempObj = tempVec.get(i);
      try
      {
         input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Physical
Component/"+tempObj.toString()));
      }
      catch(IOException ioException)
      {
         JOptionPane.showMessageDialog(this,
               "Exception during input stream creation", "Exception",
               JOptionPane.ERROR_MESSAGE);
      }
      try
      {
         tempSet = (ScenarioElementSet)input.readObject();
      }
      catch(ClassNotFoundException classnotfoundException)
      {
         JOptionPane.showMessageDialog(this,
               "Exception during file read - the object was not found",
```

```
                        "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       PhysicalComponent tempPhysComp = new PhysicalComponent();
                       tempObj = tempSet.elementContainer.get(j);
                       tempPhysComp = (PhysicalComponent)tempObj;
                       physicalComponentListVec.add(tempPhysComp.getNameTextField());
                    }
                 }
                 componentSourceList.setListData(physicalComponentListVec);
                 tempObj = tempVec = null;
                 //Network
                 tempObj = aScenario.scenarioManager.get(7);
                 tempVec = (Vector)tempObj;
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Network/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
```

```
        catch(IOException ioException)
        {
           JOptionPane.showMessageDialog(this, "Exception during file close",
                                "Exception", JOptionPane.ERROR_MESSAGE);
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
           Network tempNetwork = new Network();
           tempObj = tempSet.elementContainer.get(j);
           tempNetwork = (Network)tempObj;
           networkAppliedComboBox.addItem(tempNetwork.getNameTextField());
           networkListVec.add(tempNetwork.getNameTextField());
        }
     }
     networkSourceList.setListData(networkListVec);
     tempObj = tempVec = null;
  }

  //populateStaticSourceLists() takes a list of vectors as
  //parameters in ScenarioDefinitionTool.java and is used to add static
  //content to combo boxes and list boxes.
  //First any combo boxes are cleared and reinitialized
  //Second if the target of the vector is a combo box - for each
  //element in the vector add it to the combo box
       // if the target of the vector is a list box just add the
       // vector directly
  public void populateStaticSourceLists(java.util.Vector aFilterBlocking,
          java.util.Vector aFilterApps)
  {
     //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
     //to populate:
     //filter blocking direction
     //filter applications

     //applications filtered
     applicationSourceList.setListData(aFilterApps);
     //blocking direction
     for(int i = 0; i < aFilterBlocking.size(); i++)
     {
        directionComboBox.addItem(aFilterBlocking.get(i));
     }
  }

  //After save operations cause the list boxes in forms to be cleared
  //this method repopulates those list boxes and reinitilizes them.
  //The code in this method is literally taken line for line from the
  //button listeners above.
  public void reInitLists(java.util.Vector aFilterApps)
  {
     DefaultListModel listModel = new DefaultListModel();
     Object tempObject = new Object();
     //application lists
     applicationSourceList.setListData(aFilterApps);
     for(int i = 0; i < mApplicationVector.size(); i++)
     {
        tempObject = mApplicationVector.get(i);
        listModel.addElement(tempObject);
     }
     applicationFinalList = null;
     applicationFinalList = new JList(listModel);
     applicationFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
     applicationFinalScrollPane.setViewportView(applicationFinalList);
```

```
    //component lists
    listModel = new DefaultListModel();
    tempObject = new Object();
    componentSourceList.setListData(physicalComponentListVec);
    for(int i = 0; i < mComponentVector.size(); i++)
    {
       tempObject = mComponentVector.get(i);
       listModel.addElement(tempObject);
    }
    componentFinalList = null;
    componentFinalList = new JList(listModel);
    componentFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    componentFinalScrollPane.setViewportView(componentFinalList);

    //network lists
    listModel = new DefaultListModel();
    tempObject = new Object();
    networkSourceList.setListData(networkListVec);
    for(int i = 0; i < mNetworkVector.size(); i++)
    {
       tempObject = mNetworkVector.get(i);
       listModel.addElement(tempObject);
    }
    networkFinalList = null;
    networkFinalList = new JList(listModel);
    networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
    networkFinalScrollPane.setViewportView(networkFinalList);
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
    applicationMoveButton.addActionListener(new java.awt.event.ActionListener()
    {
       public void actionPerformed(java.awt.event.ActionEvent evt)
       {
          applicationMoveButtonActionPerformed(evt);
       }
    });

    applicationRemoveButton.addActionListener(new java.awt.event.ActionListener()
    {
       public void actionPerformed(java.awt.event.ActionEvent evt)
       {
          applicationRemoveButtonActionPerformed(evt);
       }
    });

    componentMoveButton.addActionListener(new java.awt.event.ActionListener()
    {
       public void actionPerformed(java.awt.event.ActionEvent evt)
       {
          componentMoveButtonActionPerformed(evt);
       }
    });
    componentRemoveButton.addActionListener(new java.awt.event.ActionListener()
    {
       public void actionPerformed(java.awt.event.ActionEvent evt)
       {
```

```
            componentRemoveButtonActionPerformed(evt);
        }
    });

    networkMoveButton.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            networkMoveButtonActionPerformed(evt);
        }
    });

    networkRemoveButton.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            networkRemoveButtonActionPerformed(evt);
        }
    });
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario,
     java.util.Vector aFilterBlocking, java.util.Vector aFilterApps)
{
    populateDynamicSourceLists(aPath, aScenario);
    populateStaticSourceLists(aFilterBlocking, aFilterApps);

    networkAppliedComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            networkAppliedComboBoxActionPerformed(evt);
        }
    });
}

//the following name_toString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller name_toString() methods are called from the primary
//asset toString() method.

//creates the application list formatted output
private String applicationList_toString()
{
    String applicationString = new String();
    DefaultListModel listModel = new DefaultListModel();
    if(application_added)
    {
        listModel = (DefaultListModel)applicationFinalList.getModel();
        for(int i = 0; i < listModel.getSize(); i++)
        {
            applicationString +=
            "\t\tApplication: " +
            listModel.get(i).toString() +
            " :end\n";
        }
    }
    return applicationString;
}

//creates the component list formatted output
```

```
private String componentList_toString()
{
    String componentString = new String();
    DefaultListModel listModel = new DefaultListModel();
    if(component_added)
    {
        listModel = (DefaultListModel)componentFinalList.getModel();
        for(int i = 0; i < listModel.getSize(); i++)
        {
            componentString +=
            "\t\tComponent: " +
            listModel.get(i).toString() +
            " :end\n";
        }
    }
    return componentString;
}

//creates the network list formatted output
private String networkList_toString()
{
    String networkString = new String();
    DefaultListModel listModel = new DefaultListModel();
    if(network_added)
    {
        listModel = (DefaultListModel)networkFinalList.getModel();
        for(int i = 0; i < listModel.getSize(); i++)
        {
            networkString +=
            "\t\tNetwork: " +
            listModel.get(i).toString() +
            " :end\n";
        }
    }
    return networkString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an Filter descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString +=

    "\t//Filters are used by gateway-type components (e.g., routers"+
    " and firewalls)\n\t//to deny network traffic to/from specific sources"+
    " to a list of applications.\n\t//For example, one filter might deny"+
    " outgoing FTP traffic to all external\n\t//sites. Each component"+
    " that supports filters may have zero or more\n\t//filters to"+
    " constrain traffic.\n\t//\n\t//Filters default to allowing traffic."+
    "\n\n"+
    "\tFilter:\n"+
    "\t\t//Name of network this filter applies to as its \"in\" network."+
    "\n\t\t//\n\t\t//If the gateway component is not physically connected"+
    " to its \"in\"\n\t\t//network, then this filter has no effect."+
    "\n\t\t//\n\t\t//All other networks connected to this gateway device"+
    " are treated\n\t\t//as the \"out\" networks.\n\t\t//\n"+
    "\t\tNetworkAppliedTo: "+
    networkAppliedFinalTextField.getText() +
    " :end\n\n"+
```

```
    "\t\t//Direction can be BlockIn, BlockOut, or BlockBothDirections\n"+
    "\t\tDirection: "+
    directionComboBox.getSelectedItem().toString() +
    " :end\n\n"+
    "\t\t//One or more application enumerations: sendmail, receivemail,"+
    "\n\t\t//http, or ftp\n"+
    applicationList_toString() +
    "\n\t\t//If set to yes, this rule applies to all traffic to/from "+
    "components\n\t\t//outside of the company.\n"+
    "\t\tBlockAllCompanyExternal: "+
    String.valueOf(blockAllCompanyExternalCheckBox.isSelected()) +
    " :end\n\n"+
    "\t\t//If set to yes, this rule applies to all traffic to/from "+
    "components\n\t\t//not located at the same site.\n"+
    "\t\tBlockAllSiteExternal: "+
    String.valueOf(blockAllSiteExternalCheckBox.isSelected()) +
    " :end\n\n"+
    "\t\t//If set to yes, this rule applies to all traffic to/from "+
    "components\n\t\t//not located on the same \"in\" network.\n"+
    "\t\tBlockAllNetworkExternal: "+
    String.valueOf(blockAllNetworkExternalCheckBox.isSelected()) +
    " :end\n\n"+
    "\t\t//List of zero or more specfic components for which this filter"+
    " applies\n"+
    componentList_toString() +
    "\n"+
    "\t\t//List of zero or more specfic networks for which this filter"+
    " applies\n"+
    networkList_toString() +
    "\n"+
    "\t:end//Filter\n";
    return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
    return nameTextField.getText();
}

// Variables declaration - do not modify
private javax.swing.JList applicationFinalList;
private javax.swing.JScrollPane applicationFinalScrollPane;
private javax.swing.JLabel applicationLabel;
private javax.swing.JButton applicationMoveButton;
private javax.swing.JButton applicationRemoveButton;
private javax.swing.JList applicationSourceList;
private javax.swing.JScrollPane applicationSourceScrollPane;
private javax.swing.JCheckBox blockAllCompanyExternalCheckBox;
private javax.swing.JCheckBox blockAllNetworkExternalCheckBox;
private javax.swing.JCheckBox blockAllSiteExternalCheckBox;
private javax.swing.JPanel boolPanel;
private javax.swing.JList componentFinalList;
private javax.swing.JScrollPane componentFinalScrollPane;
private javax.swing.JLabel componentLabel;
private javax.swing.JButton componentMoveButton;
private javax.swing.JButton componentRemoveButton;
private javax.swing.JList componentSourceList;
private javax.swing.JScrollPane componentSourceScrollPane;
private javax.swing.JComboBox directionComboBox;
private javax.swing.JLabel directionLabel;
```

```
    private javax.swing.JPanel filterPanel;
    private javax.swing.JScrollPane filterScrollPane;
    private javax.swing.JPanel listsPanel;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JComboBox networkAppliedComboBox;
    private javax.swing.JTextField networkAppliedFinalTextField;
    private javax.swing.JLabel networkAppliedLabel;
    private javax.swing.JList networkFinalList;
    private javax.swing.JScrollPane networkFinalScrollPane;
    private javax.swing.JLabel networkLabel;
    private javax.swing.JButton networkMoveButton;
    private javax.swing.JButton networkRemoveButton;
    private javax.swing.JList networkSourceList;
    private javax.swing.JScrollPane networkSourceScrollPane;
    private javax.swing.JPanel textAndComboPanel;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private boolean application_added;
    private boolean component_added;
    private boolean network_added;
    private Vector mNetworkVector;
    private Vector mApplicationVector;
    private Vector mComponentVector;
    private Vector physicalComponentListVec;
    private Vector networkListVec;
}
```

L. THE SOURCE OF: INTEGRITY

```
/*
 * Integrity.java
 *
 * Created on December 16, 2003, 9:53 AM
 */

package CCSDT;

/**
 *
 * @author  kjohns
 *  This class is the graphical representation of an Integrity descriptor.
 *This is where the Integrity form is managed and the Integrity
 *toString() resides.
 */
public class Integrity extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Integrity */
    public Integrity()
    {
        initComponents();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        integrityScrollPane = new javax.swing.JScrollPane();
        integrityPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        levelLabel = new javax.swing.JLabel();
```

```
levelTextField = new javax.swing.JTextField();
categoryLabel = new javax.swing.JLabel();
valueLabel = new javax.swing.JLabel();
valueTextField = new javax.swing.JTextField();
valuechangeLabel = new javax.swing.JLabel();
valuechangeTextField = new javax.swing.JTextField();
attackervalueLabel = new javax.swing.JLabel();
attackervalueTextField = new javax.swing.JTextField();
attackervaluechangeLabel = new javax.swing.JLabel();
attackervaluechangeTextField = new javax.swing.JTextField();
initialbackgroundcheckLabel = new javax.swing.JLabel();
initialbackgroundcheckComboBox = new javax.swing.JComboBox();
categoryTextField = new javax.swing.JTextField();

setLayout(new java.awt.GridLayout(1, 0));

integrityPanel.setLayout(new java.awt.GridBagLayout());

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Integrity Tag Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(nameTextField, gridBagConstraints);

levelLabel.setLabelFor(levelTextField);
levelLabel.setText("Integrity Level:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(levelLabel, gridBagConstraints);

levelTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(levelTextField, gridBagConstraints);

categoryLabel.setText("Integrity Category:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(categoryLabel, gridBagConstraints);

valueLabel.setLabelFor(valueTextField);
valueLabel.setText("Value of Accurate Data:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(valueLabel, gridBagConstraints);

valueTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
```

```
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(valueTextField, gridBagConstraints);

valuechangeLabel.setLabelFor(valuechangeTextField);
valuechangeLabel.setText("Monthly Change in integrity Value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(valuechangeLabel, gridBagConstraints);

valuechangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(valuechangeTextField, gridBagConstraints);

attackervalueLabel.setLabelFor(attackervalueTextField);
attackervalueLabel.setText("Value of Corrupt  Data for an attacker:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(attackervalueLabel, gridBagConstraints);

attackervalueTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(attackervalueTextField, gridBagConstraints);

attackervaluechangeLabel.setLabelFor(attackervaluechangeTextField);
attackervaluechangeLabel.setText("Monthly Change in Attacker Value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(attackervaluechangeLabel, gridBagConstraints);

attackervaluechangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
integrityPanel.add(attackervaluechangeTextField, gridBagConstraints);

initialbackgroundcheckLabel.setLabelFor(initialbackgroundcheckComboBox);
initialbackgroundcheckLabel.setText("Initial Background Check:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
integrityPanel.add(initialbackgroundcheckLabel, gridBagConstraints);

initialbackgroundcheckComboBox.setPreferredSize(new java.awt.Dimension(150, 25));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
    integrityPanel.add(initialbackgroundcheckComboBox,  gridBagConstraints);

    categoryTextField.setPreferredSize(new java.awt.Dimension(150, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 1;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    integrityPanel.add(categoryTextField, gridBagConstraints);

    integrityScrollPane.setViewportView(integrityPanel);

    add(integrityScrollPane);

}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
      // if the target of the vector is a list box just add the
      // vector directly
public void populateStaticSourceLists(java.util.Vector aBackgroundcheck)
{
    //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
    //to populate:
    //background check

    //base component
    for(int i = 0; i < aBackgroundcheck.size(); i++)
    {
        initialbackgroundcheckComboBox.addItem(aBackgroundcheck.get(i));
    }
}

public void Load()
{
}

public void Save()
{
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an Integrity descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString +=
    "Integrity:\n"+
```

```
    "\n\t//A text string to use as a Integrity label tag in this scenario"+
    " file\n"+
    "\tName: "+nameTextField.getText()+" :end\n\n"+
   "\t//A single value representing the Integrity value between 0 and 64\n"+
    "\tLevel: "+levelTextField.getText()+
    " :end \t//an enumerated value--0 = none\n\n"+
    "\t//an enumeration of Integrity categories, e.g., 1,4,7.  Alternately"+
    " the\n\t//value can be \"none\" \n"+
    "\tCategory: "+categoryTextField.getText()+
    " :end\n\n"+
    "\t//Integrity means what it is worth that the data is accurate"+
    " this a secret\n"+
    "\tIntegrityValue: "+valueTextField.getText()+" :end\n\n"+
    "\t//Integer factor from -10 to +10 determining how much the value"+
    " changes per\n\t//month.  -10 is max drop per month, 0 is no change"+
    " per month, +10 is max\n\t//increase per month\n"+
    "\tIntegrityValueChange: "+valuechangeTextField.getText()+" :end\n\n"+
    "\t//What is it worth for an attacker to make the data not accurate\n"+
    "\tAttackerValue: "+attackervalueTextField.getText()+" :end\n\n"+
    "\t//Integer factor from -10 to +10 determining how much the value"+
    " changes per\n\t//month.  -10 is max drop per month, 0 is no change"+
    " per month, +10 is max\n\t//increase per month\n"+
    "\tAttackerValueChange: "+attackervaluechangeTextField.getText()+
    " :end\n\n"+
    "\t//This is what type of check someone needs\n"+
    "\tInitialBackGroundCheck: "+
    initialbackgroundcheckComboBox.getSelectedItem().toString()+
    " :end\t//High,Med,Low,None\n\n"+
    ":end //of Integrity\n\n";
    return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
    return nameTextField.getText();
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(java.util.Vector aBackgroundcheck)
{
    populateStaticSourceLists(aBackgroundcheck);
}

// Variables declaration - do not modify
private javax.swing.JLabel attackervalueLabel;
private javax.swing.JTextField attackervalueTextField;
private javax.swing.JLabel attackervaluechangeLabel;
private javax.swing.JTextField attackervaluechangeTextField;
private javax.swing.JLabel categoryLabel;
private javax.swing.JTextField categoryTextField;
private javax.swing.JComboBox initialbackgroundcheckComboBox;
private javax.swing.JLabel initialbackgroundcheckLabel;
private javax.swing.JPanel integrityPanel;
private javax.swing.JScrollPane integrityScrollPane;
private javax.swing.JLabel levelLabel;
private javax.swing.JTextField levelTextField;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
```

```
    private javax.swing.JLabel valueLabel;
    private javax.swing.JTextField valueTextField;
    private javax.swing.JLabel valuechangeLabel;
    private javax.swing.JTextField valuechangeTextField;
    // End of variables declaration

}
```

M. THE SOURCE OF: NETWORK

```
/*
 * Network.java
 *
 * Created on November 22, 2003, 10:30 PM
 */

package CCSDT;
import javax.swing.JOptionPane;

/**
 *
 * @author  kjohns
 *This class is the graphical representation of a Network descriptor.
 *This is where the Network form is managed and the Network
 *toString() resides.
 */
public class Network extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Network */
    public Network()
    {
        initComponents();

    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        networkScrollPane = new javax.swing.JScrollPane();
        networkPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        netipLabel = new javax.swing.JLabel();
        ipTextField1 = new javax.swing.JTextField();
        ipTextField2 = new javax.swing.JTextField();
        ipTextField3 = new javax.swing.JTextField();
        ipTextField4 = new javax.swing.JTextField();
        isstaicCheckBox = new javax.swing.JCheckBox();

        setLayout(new java.awt.GridLayout(1, 0));

        networkPanel.setLayout(new java.awt.GridBagLayout());

        nameLabel.setText("Network Name:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        networkPanel.add(nameLabel, gridBagConstraints);

        nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
```

```
    gridBagConstraints.gridy = 0;
    gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    networkPanel.add(nameTextField, gridBagConstraints);

    netipLabel.setText("Network IP:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 1;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    networkPanel.add(netipLabel, gridBagConstraints);

    ipTextField1.setPreferredSize(new java.awt.Dimension(40, 20));
    networkPanel.add(ipTextField1, new java.awt.GridBagConstraints());

    ipTextField2.setPreferredSize(new java.awt.Dimension(40, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 2;
    gridBagConstraints.gridy = 1;
    networkPanel.add(ipTextField2, gridBagConstraints);

    ipTextField3.setMinimumSize(new java.awt.Dimension(40, 20));
    ipTextField3.setPreferredSize(new java.awt.Dimension(40, 20));
    networkPanel.add(ipTextField3, new java.awt.GridBagConstraints());

    ipTextField4.setMinimumSize(new java.awt.Dimension(40, 20));
    ipTextField4.setPreferredSize(new java.awt.Dimension(40, 20));
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    networkPanel.add(ipTextField4, gridBagConstraints);

    isstaicCheckBox.setSelected(true);
    isstaicCheckBox.setText("Player cannot add or remove this network from zones or components:");
    isstaicCheckBox.setHorizontalTextPosition(javax.swing.SwingConstants.LEADING);
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.gridx = 0;
    gridBagConstraints.gridy = 2;
    gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
    networkPanel.add(isstaicCheckBox, gridBagConstraints);

    networkScrollPane.setViewportView(networkPanel);

    add(networkScrollPane);

}
public void Save()
{

}

public void Load()
{

}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for a Network descriptor
public String toString(String aText)
{
    String outputString = new String();
```

```
        outputString =
        "Network:\n"+
        "\t//Name to use for the network within the scenario document and within"+
        " the game.\n"+
        "\tName: " +
        nameTextField.getText() +
        " :end\n" +
        "\t//Any standard IP address will work\n"+
        "\tNetID: " +
        ipTextField1.getText() +
        "." +
        ipTextField2.getText() +
        "." +
        ipTextField3.getText() +
        "." +
        ipTextField4.getText() +
        " :end\n" +
        "\t//If IsStatic is set to true it means that the player cannot add"+
        " or remove the\n\t//network from zones or components\n"+
        "\tIsStatic: " +
        String.valueOf(isstaicCheckBox.isSelected()) +
        " :end\n" +
        ":end\n\n" ;
        return outputString;
    }

    //since the name field of the form is private this method provides a way
    //to fetch the name.
    public String getNameTextField()
    {
        return nameTextField.getText();
    }

    // Variables declaration - do not modify
    private javax.swing.JTextField ipTextField1;
    private javax.swing.JTextField ipTextField2;
    private javax.swing.JTextField ipText Field3;
    private javax.swing.JTextField ipTextField4;
    private javax.swing.JCheckBox isstaicCheckBox;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JLabel netipLabel;
    private javax.swing.JPanel networkPanel;
    private javax.swing.JScrollPane networkScrollPane;
    // End of variables declaration


}
```

N. THE SOURCE OF: NETWORK CONNECTION

```
/*
 * NetworkConnection.java
 *
 * Created on January 28, 2004, 7:39 AM
 */

package CCSDT;
import java.util.*;
import javax.swing.*;
import java.io.*;
/**
```

```
 *
 * @author  kjohns
 *This class is the graphical representation of a CatalogComponent descriptor.
 *This is where the CatalogComponent form is managed and the CatalogComponent
 *toString() resides.
 */
public class NetworkConnection extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form NetworkConnection */
     public NetworkConnection()
    {
        initComponents();
        acl_added = false;
        aclTotalAccessListVec = new Vector();
        aclControlComboBox.addItem("Y");
        aclControlComboBox.addItem("N");
        aclControlComboBox.addItem("X");
        aclExecuteComboBox.addItem("Y");
        aclExecuteComboBox.addItem("N");
        aclExecuteComboBox.addItem("X");
        aclReadComboBox.addItem("Y");
        aclReadComboBox.addItem("N");
        aclReadComboBox.addItem("X");
        aclWriteComboBox.addItem("Y");
        aclWriteComboBox.addItem("N");
        aclWriteComboBox.addItem("X");
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        mlslbuttonGroup = new javax.swing.ButtonGroup();
        dacbuttonGroup = new javax.swing.ButtonGroup();
        networkConnectionsScrollPane = new javax.swing.JScrollPane();
        networkConnectionsPanel = new javax.swing.JPanel();
        connectionNameLabel = new javax.swing.JLabel();
        connectionNameTextField = new javax.swing.JTextField();
        networkNameLabel = new javax.swing.JLabel();
        networkNameComboBox = new javax.swing.JComboBox();
        networkNameFinalTextField = new javax.swing.JTextField();
        DACPanel = new javax.swing.JPanel();
        ugwPanel = new javax.swing.JPanel();
        userLabel = new javax.swing.JLabel();
        userComboBox = new javax.swing.JComboBox();
        groupLabel = new javax.swing.JLabel();
        groupComboBox = new javax.swing.JComboBox();
        modeLabel = new javax.swing.JLabel();
        userTextField = new javax.swing.JTextField();
        groupTextField = new javax.swing.JTextField();
        worldTextField = new javax.swing.JTextField();
        userFinalTextField = new javax.swing.JTextField();
        groupFinalTextField = new javax.swing.JTextField();
        aclPanel = new javax.swing.JPanel();
        aclUserLabel = new javax.swing.JLabel();
        aclGroupLabel = new javax.swing.JLabel();
        aclReadLabel = new javax.swing.JLabel();
        aclWriteLabel = new javax.swing.JLabel();
        aclControlLabel = new javax.swing.JLabel();
        aclExecuteLabel = new javax.swing.JLabel();
        aclUserComboBox = new javax.swing.JComboBox();
        aclUserFinalTextField = new javax.swing.JTextField();
```

```
aclGroupComboBox = new javax.swing.JComboBox();
aclGroupFinalTextField = new javax.swing.JTextField();
aclReadComboBox = new javax.swing.JComboBox();
aclWriteComboBox = new javax.swing.JComboBox();
aclControlComboBox = new javax.swing.JComboBox();
aclExecuteComboBox = new javax.swing.JComboBox();
aclAddButton = new javax.swing.JButton();
aclScrollPane = new javax.swing.JScrollPane();
aclList = new javax.swing.JList();
aclRemoveButton = new javax.swing.JButton();
ugwRadioButton = new javax.swing.JRadioButton();
aclRadioButton = new javax.swing.JRadioButton();
slmlPanel = new javax.swing.JPanel();
slRadioButton = new javax.swing.JRadioButton();
slSecrecyLabel = new javax.swing.JLabel();
slSecrecyComboBox = new javax.swing.JComboBox();
slSecrecyFinalTextField = new javax.swing.JTextField();
slIntegrityLabel = new javax.swing.JLabel();
slIntegrityComboBox = new javax.swing.JComboBox();
slIntegrityFinalTextField = new javax.swing.JTextField();
mlRadioButton = new javax.swing.JRadioButton();
mlMaxSecrecyLabel = new javax.swing.JLabel();
mlMaxSecrecyComboBox = new javax.swing.JComboBox();
mlMaxSecrecyFinalTextField = new javax.swing.JTextField();
mlMinSecrecyLabel = new javax.swing.JLabel();
mlMinSecrecyComboBox = new javax.swing.JComboBox();
mlMinSecrecyFinalTextField = new javax.swing.JTextField();
mlMaxIntegrityLabel = new javax.swing.JLabel();
mlM axIntegrityComboBox = new javax.swing.JComboBox();
mlMaxIntegrityFinalTextField = new javax.swing.JTextField();
mlMinIntegrityLabel = new javax.swing.JLabel();
mlMinIntegrityComboBox = new javax.swing.JComboBox();
mlMinIntegrityFinalTextField = new javax.swing.JTextField();

setLayout(new java.awt.BorderLayout());

networkConnectionsPanel.setLayout(new java.awt.GridBagLayout());

networkConnectionsPanel.setMinimumSize(new java.awt.Dimension(400, 100));
networkConnectionsPanel.setPreferredSize(new java.awt.Dimension(749, 745));
connectionNameLabel.setText("Connection Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
networkConnectionsPanel.add(connectionNameLabel, gridBagConstraints);

connectionNameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
networkConnectionsPanel.add(connectionNameTextField, gridBagConstraints);

networkNameLabel.setText("Network:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
networkConnectionsPanel.add(networkNameLabel, gridBagConstraints);

networkNameComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
networkNameComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      networkNameComboBoxActionPerformed(evt);
   }
```

```
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
networkConnectionsPanel.add(networkNameComboBox, gridBagConstraints);

networkNameFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
networkConnectionsPanel.add(networkNameFinalTextField, gridBagConstraints);

DACPanel.setLayout(new java.awt.GridBagLayout());

DACPanel.setBorder(new javax.swing.border.TitledBorder("Discretionary Access Controls"));
ugwPanel.setLayout(new java.awt.GridBagLayout());

ugwPanel.setBorder(new javax.swing.border.TitledBorder("User Group World"));
userLabel.setText("User:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
ugwPanel.add(userLabel, gridBagConstraints);

userComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
userComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      userComboBoxActionPerformed(evt);
   }
});

ugwPanel.add(userComboBox, new java.awt.GridBagConstraints());

groupLabel.setText("Group:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
ugwPanel.add(groupLabel, gridBagConstraints);

groupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
groupComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      groupComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
ugwPanel.add(groupComboBox, gridBagConstraints);

modeLabel.setText("Permissions:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
ugwPanel.add(modeLabel, gridBagConstraints);

userTextField.setPreferredSize(new java.awt.Dimension(50, 20));
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
ugwPanel.add(userTextField, gridBagConstraints);

groupTextField.setPreferredSize(new java.awt.Dimension(50, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
ugwPanel.add(groupTextField, gridBagConstraints);

worldTextField.setPreferredSize(new java.awt.Dimension(50, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
ugwPanel.add(worldTextField, gridBagConstraints);

userFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
ugwPanel.add(userFinalTextField, new java.awt.GridBagConstraints());

groupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
ugwPanel.add(groupFinalTextField, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
DACPanel.add(ugwPanel, gridBagConstraints);

aclPanel.setLayout(new java.awt.GridBagLayout());

aclPanel.setBorder(new javax.swing.border.TitledBorder("Access Control List"));
aclUserLabel.setText("User");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
aclPanel.add(aclUserLabel, gridBagConstraints);

aclGroupLabel.setText("Group");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
aclPanel.add(aclGroupLabel, gridBagConstraints);

aclReadLabel.setText("Read");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
aclPanel.add(aclReadLabel, gridBagConstraints);

aclWriteLabel.setText("Write");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
aclPanel.add(aclWriteLabel, gridBagConstraints);
```

```
aclControlLabel.setText("Control");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 3;
aclPanel.add(aclControlLabel, gridBagConstraints);

aclExecuteLabel.setText("Execute");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
aclPanel.add(aclExecuteLabel, gridBagConstraints);

aclUserComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclUserComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclUserComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
aclPanel.add(aclUserComboBox, gridBagConstraints);

aclUserFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
aclPanel.add(aclUserFinalTextField, gridBagConstraints);

aclGroupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclGroupComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
aclPanel.add(aclGroupComboBox, gridBagConstraints);

aclGroupFinalText Field.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
aclPanel.add(aclGroupFinalTextField, gridBagConstraints);

aclReadComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
aclPanel.add(aclReadComboBox, gridBagConstraints);

aclWriteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
aclPanel.add(aclWriteComboBox, gridBagConstraints);

aclControlComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 4;
aclPanel.add(aclControlComboBox, gridBagConstraints);

aclExecuteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 4;
aclPanel.add(aclExecuteComboBox, gridBagConstraints);

aclAddButton.setText("Add");
aclAddButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclAddButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclAddButton, gridBagConstraints);

aclScrollPane.setPreferredSize(new java.awt.Dimension(700, 150));
aclScrollPane.setViewportView(aclList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclScrollPane, gridBagConstraints);

aclRemoveButton.setText("Remove");
aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclRemoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclRemoveButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
DACPanel.add(aclPanel, gridBagConstraints);

ugwRadioButton.setSelected(true);
ugwRadioButton.setText("Use permission bits");
dacbuttonGroup.add(ugwRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
DACPanel.add(ugwRadioButton, gridBagConstraints);
```

```
aclRadioButton.setText("Use access control lists");
dacbuttonGroup.add(aclRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
DACPanel.add(aclRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
networkConnectionsPanel.add(DACPanel, gridBagConstraints);

slmlPanel.setLayout(new java.awt.GridBagLayout());

slmlPanel.setBorder(new javax.swing.border.TitledBorder("MAC Connection Settings"));
slRadioButton.setSelected(true);
slRadioButton.setText("Single Level Connection");
mlslbuttonGroup.add(slRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(slRadioButton, gridBagConstraints);

slSecrecyLabel.setText("Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(slSecrecyLabel, gridBagConstraints);

slSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
slSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      slSecrecyComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(slSecrecyComboBox, gridBagConstraints);

slSecrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
slmlPanel.add(slSecrecyFinalTextField, gridBagConstraints);

slIntegrityLabel.setText("Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(slIntegrityLabel, gridBagConstraints);

slIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
slIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      slIntegrityComboBoxActionPerformed(evt);
   }
```

```
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(slIntegrityComboBox, gridBagConstraints);

slIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
slmlPanel.add(slIntegrityFinalTextField, gridBagConstraints);

mlRadioButton.setText("Multilevel Connection");
mlslbuttonGroup.add(mlRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(mlRadioButton, gridBagConstraints);

mlMaxSecrecyLabel.setText("Max Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(mlMaxSecrecyLabel, gridBagConstraints);

mlMaxSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMaxSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMaxSecrecyComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(mlMaxSecrecyComboBox, gridBagConstraints);

mlMaxSecrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
slmlPanel.add(mlMaxSecrecyFinalTextField, gridBagConstraints);

mlMinSecrecyLabel.setText("Min Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(mlMinSecrecyLabel, gridBagConstraints);

mlMinSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMinSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMinSecrecyComboBoxActionPerformed(evt);
   }
});
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(mlMinSecrecyComboBox, gridBagConstraints);

mlMinSecrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 5;
slmlPanel.add(mlMinSecrecyFinalTextField, gridBagConstraints);

mlMaxIntegrityLabel.setText("Max Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(mlMaxIntegrityLabel, gridBagConstraints);

mlMaxIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMaxIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMaxIntegrityComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(mlMaxIntegrityComboBox, gridBagConstraints);

mlMaxIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 6;
slmlPanel.add(mlMaxIntegrityFinalTextField, gridBagConstraints);

mlMinIntegrityLabel.setText("Min Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
slmlPanel.add(mlM inIntegrityLabel, gridBagConstraints);

mlMinIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMinIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMinIntegrityComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
slmlPanel.add(mlMinIntegrityComboBox, gridBagConstraints);

mlMinIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 7;
```

```
        slmlPanel.add(mlMinIntegrityFinalTextField, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        networkConnectionsPanel.add(slmlPanel, gridBagConstraints);

        networkConnectionsScrollPane.setViewportView(networkConnectionsPanel);

        add(networkConnectionsScrollPane, java.awt.BorderLayout.CENTER);

    }
//the following listeners set a text field based on a combo box selection
    private void networkNameComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(networkNameComboBox.getSelectedItem() != null)
        {
            networkNameFinalTextField.setText(
                    networkNameComboBox.getSelectedItem().toString());
        }
    }

    private void mlMinIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMinIntegrityComboBox.getSelectedItem() != null)
        {
            mlMinIntegrityFinalTextField.setText(
                    mlMinIntegrityComboBox.getSelectedItem().toString());
        }
    }

    private void mlMaxIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMaxIntegrityComboBox.getSelectedItem() != null)
        {
            mlMaxIntegrityFinalTextField.setText(
                    mlMaxIntegrityComboBox.getSelectedItem().toString());
        }
    }

    private void mlMinSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMinSecrecyComboBox.getSelectedItem() != null)
        {
            mlMinSecrecyFinalTextField.setText(
                    mlMinSecrecyComboBox.getSelectedItem().toString());
        }
    }

    private void mlMaxSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMaxSecrecyComboBox.getSelectedItem() != null)
        {
            mlMaxSecrecyFinalTextField.setText(
                    mlMaxSecrecyComboBox.getSelectedItem().toString());
        }
    }

    private void slIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(slIntegrityComboBox.getSelectedItem() != null)
        {
            slIntegrityFinalTextField.setText(
                    slIntegrityComboBox.getSelectedItem().toString());
        }
    }
```

```
    private void slSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(slSecrecyComboBox.getSelectedItem() != null)
        {
            slSecrecyFinalTextField.setText(
                        slSecrecyComboBox.getSelectedItem().toString());
        }
    }

    private void groupComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(groupComboBox.getSelectedItem() != null)
        {
            groupFinalTextField.setText(
                            groupComboBox.getSelectedItem().toString());
        }
    }

    private void userComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(userComboBox.getSelectedItem() != null)
        {
            userFinalTextField.setText(
                            userComboBox.getSelectedItem().toString());
        }
    }

    private void aclGroupComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclGroupComboBox.getSelectedItem() != null)
        {
            aclGroupFinalTextField.setText(
                        aclGroupComboBox.getSelectedItem().toString());
        }
    }

    private void aclUserComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclUserComboBox.getSelectedItem() != null)
        {
            aclUserFinalTextField.setText(
                        aclUserComboBox.getSelectedItem().toString());
        }
    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
    private void aclRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(aclList.getSelectedIndex() >= 0)
        {
            aclTotalAccessListVec.remove(aclList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)aclList.getModel();
            listModel.remove(aclList.getSelectedIndex());
            aclList = new JList(listModel);
            aclList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            aclScrollPane.setViewportView(aclList);
        }
    }

    private void aclAddButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        mACL = new AccessControlList();
        Object tempObject = new Object();
        AccessControlList tempACL = new AccessControlList();
        mACL.setUser(aclUserFinalTextField.getText());
```

```
   mACL.setGroup(aclGroupFinalTextField.getText());
   mACL.setRead(aclReadComboBox.getSelectedItem().toString());
   mACL.setWrite(aclWriteComboBox.getSelectedItem().toString());
   mACL.setControl(aclControlComboBox.getSelectedItem().toString());
   mACL.setExecute(aclExecuteComboBox.getSelectedItem().toString());
   aclTotalAccessListVec.addElement(mACL);
   DefaultListModel listModel = new DefaultListModel();
   for(int i = 0; i < aclTotalAccessListVec.size(); i++)
   {
      tempObject = aclTotalAccessListVec.get(i);
      tempACL = (AccessControlList)tempObject;
      listModel.addElement(tempACL.getACL());
   }
   aclList = null;
   aclList = new JList(listModel);
   aclList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   aclScrollPane.setViewportView(aclList);
   acl_added = true;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
   return connectionNameTextField.getText();
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateDynamicSourceLists() takes datapath and startupScenario as
//parameters in ScenarioDefinitionTool.java and is used to add dynamic
//content to combo boxes and list boxes.
//First the scenairo is used to get the added reusable sets
//Second any combo boxes to be used are cleared and reinitialized
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
{
   //uses the Scenario passed to it to set the dynamic values of
   //dependent sets
   //Network
   //User
   //DACGroup
   //Secrecy
   //Integrity
   mScenario = aScenario;
   Object tempObj = new Object();
   Vector tempVec = new Vector();
   ScenarioElementSet tempSet = new ScenarioElementSet();
   //Network
   tempObj = aScenario.scenarioManager.get(7);
   tempVec = (Vector)tempObj;
   networkNameComboBox.removeAllItems();
```

```
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Network/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   for(int j = 0; j < tempSet.elementContainer.size(); j++)
                   {
                      Network tempNetwork = new Network();
                      tempObj = tempSet.elementContainer.get(j);
                      tempNetwork = (Network)tempObj;
                      networkNameComboBox.addItem(tempNetwork.getNameTextField());
                   }
                }
                //User
                tempObj = aScenario.scenarioManager.get(11);
                tempVec = (Vector)tempObj;
                aclUserComboBox.removeAllItems();
                userComboBox.removeAllItems();
                aclUserComboBox.addItem("*");
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
```

```
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                    User tempUser = new User();
                    tempObj = tempSet.elementContainer.get(j);
                    tempUser = (User)tempObj;
                    aclUserComboBox.addItem(tempUser.getNameTextField());
                    userComboBox.addItem(tempUser.getNameTextField());
                }
            }
            //DACGroup
            tempObj = aScenario.scenarioManager.get(4);
            tempVec = (Vector)tempObj;
            aclGroupComboBox.removeAllItems();
            groupComboBox.removeAllItems();
            aclGroupComboBox.addItem("*");
            for(int i = 0; i < tempVec.size(); i++)
            {
                tempObj = tempVec.get(i);
                try
                {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
```

```
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       DACGroup tempDAC = new DACGroup();
                       tempObj = tempSet.elementContainer.get(j);
                       tempDAC = (DACGroup)tempObj;
                       aclGroupComboBox.addItem(tempDAC.getNameTextField());
                       groupComboBox.addItem(tempDAC.getNameTextField());
                    }
                 }
                 //Secrecy
                 tempObj = aScenario.scenarioManager.get(9);
                 tempVec = (Vector)tempObj;
                 slSecrecyComboBox.removeAllItems();
                 mlMaxSecrecyComboBox.removeAllItems();
                 mlMinSecrecyComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during input stream creation", "Exception",
                                JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
```

```
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       Secrecy tempSecrecy = new Secrecy();
                       tempObj = tempSet.elementContainer.get(j);
                       tempSecrecy = (Secrecy)tempObj;
                       slSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                       mlMaxSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                       mlMinSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                    }
                 }
                 //Integrity
                 slIntegrityComboBox.removeAllItems();
                 mlMaxIntegrityComboBox.removeAllItems();
                 mlMinIntegrityComboBox.removeAllItems();
                 tempObj = aScenario.scenarioManager.get(6);
                 tempVec = (Vector)tempObj;
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
```

```
                                "Exception", JOptionPane.ERROR_MESSAGE);
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
           Integrity tempIntegrity = new Integrity();
           tempObj = tempSet.elementContainer.get(j);
           tempIntegrity = (Integrity)tempObj;
           slIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
           mlMaxIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
           mlMinIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
        }
     }
}

public void Load()
{
}

public void Save()
{
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
   DefaultListModel listModel = new DefaultListModel();
   Object tempObject = new Object();
   AccessControlList tempACL = new AccessControlList();
   for(int i = 0; i < aclTotalAccessListVec.size(); i++)
   {
      tempObject = aclTotalAccessListVec.get(i);
      tempACL = (AccessControlList)tempObject;
      listModel.addElement(tempACL.getACL());
   }
   aclList = null;
   aclList = new JList(listModel);
   aclList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   aclScrollPane.setViewportView(aclList);
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
   aclAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclAddButtonActionPerformed(evt);
      }
   });

   aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclRemoveButtonActionPerformed(evt);
      }
   });
}
```

```
//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario)
{
   populateDynamicSourceLists(aPath, aScenario);

   networkNameComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         networkNameComboBoxActionPerformed(evt);
      }
   });

   aclUserComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclUserComboBoxActionPerformed(evt);
      }
   });

   aclGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclGroupComboBoxActionPerformed(evt);
      }
   });

   userComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         userComboBoxActionPerformed(evt);
      }
   });

   groupComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         groupComboBoxActionPerformed(evt);
      }
   });

   slSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         slSecrecyComboBoxActionPerformed(evt);
      }
   });

   slIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         slIntegrityComboBoxActionPerformed(evt);
      }
   });

   mlMaxSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMaxSecrecyComboBoxActionPerformed(evt);
      }
   });

   mlMinSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMinSecrecyComboBoxActionPerformed(evt);
      }
   });

   mlMaxIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
```

```
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            mlMaxIntegrityComboBoxActionPerformed(evt);
        }
    });

    mlMinIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            mlMinIntegrityComboBoxActionPerformed(evt);
        }
    });
}

//the following name_toString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller name_toString() methods are called from the primary
//asset toString() method.

//creates the ACL list formatted output
private String aclList_toString()
{
    Object tempObject = new Object();
    AccessControlList tempACL = new AccessControlList();
    String aclString = new String();
    if(acl_added)
    {
        for(int i = 0; i < aclTotalAccessListVec.size(); i++)
        {
            tempObject = aclTotalAccessListVec.get(i);
            tempACL = (AccessControlList)tempObject;
            aclString +=
            "\t\tAccessList: " +
            tempACL.getUser() + "." + tempACL.getGroup()+
            " :end\tAccessMode: " +
            tempACL.getRead() + tempACL.getWrite() +
            tempACL.getClass() + tempACL.getExecute() +
            " :end\n";
        }
    }
    return aclString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for a NetworkConnection descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString = "\n\t//List of networks connected to this component."+
    "\n\tNetwork: " +
    "\n\t\t//the tag name of a Network defined in this SDF\n"+
    "\t\tName: "+
    networkNameFinalTextField.getText() +
    " :end\n\n";
    if(ugwRadioButton.isSelected() || aclRadioButton.isSelected())
    {
        outputString += "\t\t//The DAC applied to this network connection (if any).  If the"+
        "component\n\t\t//is instaniated (i.e., \"IsTemplate\" is false),"+
        " then\n\t\t//this is the initial DAC on the connection when the "+
        "game begins.  If the\n\t\t//component is a template, then the DAC"+
        " settings can be changed when\n\t\t//the player obtains the component"+
        " based on previous player choices.  \n\t\t//The DAC can be either a"+
```

```
   "\"user group world\" specification or an access\n\t\t//control list"+
   " (ACL) depending on what is supported by the component.\n\t\t//The"+
   " ACL format here is similar to the format in Assets but there is\n"+
   "\t\t//no $ symbol to relfect an asset created by the user."+
   "\n\t\t//(*|%name%).(*|%group%) %read% %write% %control% %execute%\n\n"+
   "\t\t//Access list entry.  Available modes are Y=Yes N=No and X=Don't"+
   " care\n";
   if(aclRadioButton.isSelected())
   {
      outputString += aclList_toString();
   }
   if(ugwRadioButton.isSelected())
   {
      outputString +="\n\t\tUserGroupWorld:\n\t\t\t//identifies the user and group that owns"+
      " the asset.  Wildcards are\n\t\t\t//not permitted.\n"+
      "\t\t\tUser: "+
      userFinalTextField.getText() +
      " :end\n"+
      "\t\t\tGroup: "+
      groupFinalTextField.getText() +
      " :end\n"+
      "\t\t\tMode: "+
      userTextField.getText() + groupTextField.getText() +
      worldTextField.getText() +
      " :end //Unix style mode\n\t\t:end //of UserGroupWorld:\n\n";
   }
}
if(this.slRadioButton.isSelected() || this.mlRadioButton.isSelected())
{
   outputString += "\t\t//Mandatory access controls over access to the network"+
   " connection\n\t\t//(if any). This is ignored if the component's"+
   " OS does not enforce a\n\t\t//MAC policy.  If ths comp onent is a"+
   " template, the MAC setting can be\n\t\t//over-ridden by player"+
   "choice made before acquiring the component.\n\t\t//Values are tag "+
   "names from Secrecy Label and Integrity Label entries\n\t\t//from"+
   " this SDF.";
   if(this.slRadioButton.isSelected())
   {
      outputString += "\n\n\t\t//Single level connection specification\n"+
      "\t\tSLConnection:\n"+
      "\t\t\tSecrecyLabel: "+
      slSecrecyFinalTextField.getText() +
      " :end\n"+
      "\t\t\tIntegrityLabel: "+
      slIntegrityFinalTextField.getText()+
      " :end\n"+
      "\t\t:end //of SLConnection\n\n";
   }
   if(this.mlRadioButton.isSelected())
   {
      outputString += "\t\t//multilevel connection "+
      "specification.  This is mutually exclusive with the\n\t\t//"+
      "Single level connection specification, and is only meaningful if"+
      "the\n\t\t//component's OS supports multilevel connections.\n"+
      "\t\tMLConnection:\n"+
      "\t\t\tMaxSecrecyLabel: "+
      mlMaxSecrecyFinalTextField.getText()+
      " :end\n"+
      "\t\t\tMinSecrecyLabel: "+
      mlMinSecrecyFinalTextField.getText()+
      " :end\n"+
      "\t\t\tMaxIntegrityLabel: "+
```

```
            mlMaxIntegrityFinalTextField.getText()+
            " :end\n"+
            "\t\t\tMinIntegrityLabel: "+
            mlMinIntegrityFinalTextField.getText()+
            " :end\n"+
            "\t\t:end //of MLConnection\n";
        }
    }
    outputString += "\t:end//of network description\n\n";
    return outputString;
}

// Variables declaration - do not modify
private javax.swing.JPanel DACPanel;
private javax.swing.JButton aclAddButton;
private javax.swing.JComboBox aclControlComboBox;
private javax.swing.JLabel aclControlLabel;
private javax.swing.JComboBox aclExecuteComboBox;
private javax.swing.JLabel aclExecuteLabel;
private javax.swing.JComboBox aclGroupComboBox;
private javax.swing.JTextField aclGroupFinalTextField;
private javax.swing.JLabel aclGroupLabel;
private javax.swing.JList aclList;
private javax.swing.JPanel aclPanel;
private javax.swing.JRadioButton aclRadioButton;
private javax.swing.JComboBox aclReadComboBox;
private javax.swing.JLabel aclReadLabel;
private javax.swing.JButton aclRemoveButton;
private javax.swing.JScrollPane aclScrollPane;
private javax.swing.JComboBox aclUserComboBox;
private javax.swing.JTextField aclUserFinalTextField;
private javax.swing.JLabel aclUserLabel;
private javax.swing.JComboBox aclWriteComboBox;
private javax.swing.JLabel aclWriteLabel;
private javax.swing.JLabel connectionNameLabel;
private javax.swing.JTextField connectionNameTextField;
private javax.swing.ButtonGroup dacbuttonGroup;
private javax.swing.JComboBox groupComboBox;
private javax.swing.JTextField groupFinalTextField;
private javax.swing.JLabel groupLabel;
private javax.swing.JTextField groupTextField;
private javax.swing.JComboBox mlMaxIntegrityComboBox;
private javax.swing.JTextField mlMaxIntegrityFinalTextField;
private javax.swing.JLabel mlMaxIntegrityLabel;
private javax.swing.JComboBox mlMaxSecrecyComboBox;
private javax.swing.JTextField mlMaxSecrecyFinalTextField;
private javax.swing.JLabel mlMaxSecrecyLabel;
private javax.swing.JComboBox mlMinIntegrityComboBox;
private javax.swing.JTextField mlMinIntegrityFinalTextField;
private javax.swing.JLabel mlMinIntegrityLabel;
private javax.swing.JComboBox mlMinSecrecyComboBox;
private javax.swing.JTextField mlMinSecrecyFinalTextField;
private javax.swing.JLabel mlMinSecrecyLabel;
private javax.swing.JRadioButton mlRadioButton;
private javax.swing.ButtonGroup mlslbuttonGroup;
private javax.swing.JLabel modeLabel;
private javax.swing.JPanel networkConnectionsPanel;
private javax.swing.JScrollPane networkConnectionsScrollPane;
private javax.swing.JComboBox networkNameComboBox;
private javax.swing.JTextField networkNameFinalTextField;
private javax.swing.JLabel networkNameLabel;
private javax.swing.JComboBox slIntegrityComboBox;
```

```
    private javax.swing.JTextField slIntegrityFinalTextField;
    private javax.swing.JLabel slIntegrityLabel;
    private javax.swing.JRadioButton slRadioButton;
    private javax.swing.JComboBox slSecrecyComboBox;
    private javax.swing.JTextField slSecrecyFinalTextField;
    private javax.swing.JLabel slSecrecyLabel;
    private javax.swing.JPanel slmlPanel;
    private javax.swing.JPanel ugwPanel;
    private javax.swing.JRadioButton ugwRadioButton;
    private javax.swing.JComboBox userComboBox;
    private javax.swing.JTextField userFinalTextField;
    private javax.swing.JLabel userLabel;
    private javax.swing.JTextField userTextField;
    private javax.swing.JTextField worldTextField;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private Scenario mScenario;
    private boolean acl_added;
    private AccessControlList mACL;
    private Vector aclTotalAccessListVec;
}
```

O. THE SOURCE OF: OBJECT NODE

```
/*
 * ObjectNode.java
 *
 * Created on December 19, 2003, 1:07 PM
 */

package CCSDT;
import javax.swing.tree.*;

/**
 *
 * @author  kjohns
 *
 *This class iteratively explores a Vector and uses the results to
 *populate a tree.
 */
public class ObjectNode extends DefaultMutableTreeNode
{

    /** Creates a new instance of ObjectNode */
    public ObjectNode(Scenario aObject)
    {
        setUserObject(aObject);
    }

    public DefaultTreeModel explore()
    {
        //Current Scenario: blah (root)
            //Network (majorset)
                //BubbaNet (subset)
                //GumpNet (subset)
                //ShrimpNet (subset)
            //Secrecy (majorset)
                //SCISec (subset)
                //TSSec (subset)
            //Integrity (majorset)
                //TrustedInt (subset)
        //...
```

```
                Object tempObject = getUserObject();
                Object Object_f = getUserObject();
                Object Object_p = getUserObject();
                Object Object_n = getUserObject();
                Scenario workingScenario = (Scenario)tempObject;
                root = new DefaultMutableTreeNode("Current Scenario: " + workingScenario.getScenarioFileName());
                thisTree = new DefaultTreeM odel(root);
                int scenariomangerLength = workingScenario.scenarioManager.size();
                if(workingScenario.scenarioManager.size() > 0)
                {
                   for(int a = 0; a < scenariomangerLength; a++)//for each majorset
                   {
                      if(a == 0)//Asset
                      {
                         majorset = new DefaultMutableTreeNode("Asset");//create and set the major set
                         root.add(majorset);//add the major set
                         Object tempAssetObj = workingScenario.scenarioManager.get(0);//the asset vector from
scenanrio manager
                         java.util.Vector tempAssetSet = (java.util.Vector)tempAssetObj;//cast the object to a vector
                         //becuase that is what it really is
                         int assetsetSize = tempAssetSet.size();//get the set size
                         for(int b = 0; b < assetsetSize; b++)//for each set in asset set
                         {
                            Object internalSetObj = tempAssetSet.get(b);//get the form from the set
                            String internalSetName = (String)internalSetObj;//get the forms name
                            subset = new DefaultMutableTreeNode(internalSetName);//create and set the subset
                            majorset.add(subset);//add the subset
                         }
                      }
                      if(a == 1)//asset goal
                      {
                         majorset = new DefaultMutableTreeNode("Goal");
                         root.add(majorset);
                         Object tempAssetGoalObj = workingScenario.scenarioManager.get(1);//the asset goal vector
from scenanrio manager
                         java.util.Vector tempAssetGoalSet = (java.util.Vector)tempAssetGoalObj;
                         int assetgoalsetSize = tempAssetGoalSet.size();
                         for(int b = 0; b < assetgoalsetSize; b++)//for each set in asset goal set
                         {
                            Object internalSetObj = tempAssetGoalSet.get(b);
                            String internalSetName = (String)internalSetObj;
                            subset = new DefaultMutableTreeNode(internalSetName);
                            majorset.add(subset);
                         }
                      }
                      if(a == 2)//catalog component
                      {
                         majorset = new DefaultMutableTreeNode("Catalog Component");
                         root.add(majorset);
                         Object tempCatalogObj = workingScenario.scenarioManager.get(2);//the catalog component
vector from scenanrio manager
                         java.util.Vector tempCatalogSet = (java.util.Vector)tempCatalogObj;
                         int catalogsetSize = tempCatalogSet.size();
                         for(int b = 0; b < catalogsetSize; b++)//for each set in catalog set
                         {
                            Object internalSetObj = tempCatalogSet.get(b);
                            String internalSetName = (String)internalSetObj;
                            subset = new DefaultMutableTreeNode(internalSetName);
                            majorset.add(subset);
                         }
                      }
                      if(a == 3)//Condition
```

```
                    {
                       majorset = new DefaultMutableTreeNode("Condition");
                       root.add(majorset);
                       Object tempConditionObj = workingScenario.scenarioManager.get(3);//the condition vector
from scenanrio manager
                       java.util.Vector tempConditionSet = (java.util.Vector)tempConditionObj;
                       int conditionsetSize = tempConditionSet.size();
                       for(int b = 0; b < conditionsetSize; b++)//for each set in catalog set
                       {
                          Object internalSetObj = tempConditionSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 4)//DAC Group
                    {
                       majorset = new DefaultMutableTreeNode("DAC Group");
                       root.add(majorset);
                       Object tempDACGroupObj = workingScenario.scenarioManager.get(4);//the dac group vector
from scenanrio manager
                       java.util.Vector tempDACGroupSet = (java.util.Vector)tempDACGroupObj;
                       int dacgroupsetSize = tempDACGroupSet.size();
                       for(int b = 0; b < dacgroupsetSize; b++)//for each set in catalog set
                       {
                          Object internalSetObj = tempDACGroupSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 5)//department
                    {
                       majorset = new DefaultMutableTreeNode("Department");
                       root.add(majorset);
                       Object tempDepartmentObj = workingScenario.scenarioManager.get(5);//the department vector
from scenanrio manager
                       java.util.Vector tempDepartmentSet = (java.util.Vector)tempDepartmentObj;
                       int departmentsetSize = tempDepartmentSet.size();
                       for(int b = 0; b < departmentsetSize; b++)//for each set in catalog set
                       {
                          Object internalSetObj = tempDepartmentSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 6)//integrity
                    {
                       majorset = new DefaultMutableTreeNode("Integrity");
                       root.add(majorset);
                       Object tempIntObj = workingScenario.scenarioManager.get(6);//the integrity vector from
scenanrio manager
                       java.util.Vector tempIntSet = (java.util.Vector)tempIntObj;
                       int integritysetSize = tempIntSet.size();
                       for(int b = 0; b < integritysetSize; b++)//for each set in network
                       {
                          Object internalSetObj = tempIntSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
```

```
                    }
                    if(a == 7)//network
                    {
                       majorset = new DefaultMutableTreeNode("Network");
                       root.add(majorset);
                       Object tempNetObj = workingScenario.scenarioManager.get(7);//the network vector from
scenanrio manager
                       java.util.Vector tempNetSet = (java.util.Vector)tempNetObj;
                       int networksetSize = tempNetSet.size();
                       for(int b = 0; b < networksetSize; b++)//for each set in network
                       {
                          Object internalSetObj = tempNetSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 8)//physical component
                    {
                       majorset = new DefaultMutableTreeNode("Physical Component");
                       root.add(majorset);
                       Object tempPhysicalObj = workingScenario.scenarioManager.get(8);//the physical component
vector from scenanrio manager
                       java.util.Vector tempPhysicalSet = (java.util.Vector)tempPhysicalObj;
                       int physicalsetSize = tempPhysicalSet.size();
                       for(int b = 0; b < physicalsetSize; b++)//for each set in network
                       {
                          Object internalSetObj = tempPhysicalSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 9)//secrecy
                    {
                       majorset = new DefaultMutableTreeNode("Secrecy");
                       root.add(majorset);
                       Object tempSecObj = workingScenario.scenarioManager.get(9);//the secrecy vector from
scenanrio manager
                       java.util.Vector tempSecSet = (java.util.Vector)tempSecObj;
                       int secrecysetSize = tempSecSet.size();
                       for(int b = 0; b < secrecysetSize; b++)//for each set in network
                       {
                          Object internalSetObj = tempSecSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
                          majorset.add(subset);
                       }
                    }
                    if(a == 10)//trigger
                    {
                       majorset = new DefaultMutableTreeNode("Trigger");
                       root.add(majorset);
                       Object tempTriggerObj = workingScenario.scenarioManager.get(10);//the trigger vector from
scenanrio manager
                       java.util.Vector tempTriggerSet = (java.util.Vector)tempTriggerObj;
                       int triggersetSize = tempTriggerSet.size();
                       for(int b = 0; b < triggersetSize; b++)//for each set in catalog set
                       {
                          Object internalSetObj = tempTriggerSet.get(b);
                          String internalSetName = (String)internalSetObj;
                          subset = new DefaultMutableTreeNode(internalSetName);
```

```
                        majorset.add(subset);
                    }
                }
                if(a == 11)//user
                {
                    majorset = new DefaultMutableTreeNode("User");
                    root.add(majorset);
                    Object tempUserObj = workingScenario.scenarioManager.get(11);//the User vector from
scenanrio manager
                    java.util.Vector tempUserSet = (java.util.Vector)tempUserObj;
                    int usersetSize = tempUserSet.size();
                    for(int b = 0; b < usersetSize; b++)//for each set in catalog set
                    {
                        Object internalSetObj = tempUserSet.get(b);
                        String internalSetName = (String)internalSetObj;
                        subset = new DefaultMutableTreeNode(internalSetName);
                        majorset.add(subset);
                    }
                }
                if(a == 12)//workspace
                {
                    majorset = new DefaultMutableTreeNode("Workspace");
                    root.add(majorset);
                    Object tempWorkspaceObj = workingScenario.scenarioManager.get(12);//the workspace vector
from scenanrio manager
                    java.util.Vector tempWorkspaceSet = (java.util.Vector)tempWorkspaceObj;
                    int workspacesetSize = tempWorkspaceSet.size();
                    for(int b = 0; b < workspacesetSize; b++)//for each set in catalog set
                    {
                        Object internalSetObj = tempWorkspaceSet.get(b);
                        String internalSetName = (String)internalSetObj;
                        subset = new DefaultMutableTreeNode(internalSetName);
                        majorset.add(subset);
                    }
                }
                if(a == 13)//zone
                {
                    majorset = new DefaultMutableTreeNode("Zone");
                    root.add(majorset);
                    Object tempZoneObj = workingScenario.scenarioManager.get(13);//the zone vector from
scenanrio manager
                    java.util.Vector tempZoneSet = (java.util.Vector)tempZoneObj;
                    int zonesetSize = tempZoneSet.size();
                    for(int b = 0; b < zonesetSize; b++)//for each set in network
                    {
                        Object internalSetObj = tempZoneSet.get(b);
                        String internalSetName = (String)internalSetObj;
                        subset = new DefaultMutableTreeNode(internalSetName);
                        majorset.add(subset);
                    }
                }
                if(a == 14)//other
                {
                    majorset = new DefaultMutableTreeNode("Other");
                    root.add(majorset);
                    Object tempOtherObj = workingScenario.scenarioManager.get(14);//the other vector from
scenanrio manager
                    java.util.Vector tempOtherSet = (java.util.Vector)tempOtherObj;
                    Object_f = tempOtherSet.get(0);//filter
                    java.util.Vector tempOtherSet_f = (java.util.Vector)Object_f;
                    filtersubset = new DefaultMutableTreeNode("Filter");
                    majorset.add(filtersubset);
```

```
                Object_p = tempOtherSet.get(1);//procedural settings
                java.util.Vector tempOtherSet_p = (java.util.Vector)Object_p;
                proceduresubset = new DefaultMutableTreeNode("Procedural Settings");
                majorset.add(proceduresubset);
                Object_n = tempOtherSet.get(2);//network connection
                java.util.Vector tempOtherSet_n = (java.util.Vector)Object_n;
                networksubset = new DefaultMutableTreeNode("Component Network Connection");
                majorset.add(networksubset);
                int fsetSize = tempOtherSet_f.size();
                int psetSize = tempOtherSet_p.size();
                int nsetSize = tempOtherSet_n.size();
                for(int b = 0; b < fsetSize; b++)//for each set in filter
                {
                   Object internalSetObj = tempOtherSet_f.get(b);
                   String internalSetName = (String)internalSetObj;
                   subset1 = new DefaultMutableTreeNode(internalSetName);
                   filtersubset.add(subset1);
                }
                for(int c = 0; c < psetSize; c++)//for each set in procedural
                {
                   Object internalSetObj = tempOtherSet_p.get(c);
                   String internalSetName = (String)internalSetObj;
                   subset2 = new DefaultMutableTreeNode(internalSetName);
                   proceduresubset.add(subset2);
                }
                for(int d = 0; d < nsetSize; d++)//for each set in network connection
                {
                   Object internalSetObj = tempOtherSet_n.get(d);
                   String internalSetName = (String)internalSetObj;
                   subset3 = new DefaultMutableTreeNode(internalSetName);
                   networksubset.add(subset3);
                }
             }
          }
       }

       return thisTree;
    }

    private DefaultTreeModel thisTree;
    private DefaultMutableTreeNode root;
    private DefaultMutableTreeNode majorset;
    private DefaultMutableTreeNode subset;
    private DefaultMutableTreeNode filtersubset;
    private DefaultMutableTreeNode proceduresubset;
    private DefaultMutableTreeNode networksubset;
    private DefaultMutableTreeNode subset1;
    private DefaultMutableTreeNode subset2;
    private DefaultMutableTreeNode subset3;
}
```

P. THE SOURCE OF: PHYSICAL COMPONENT

```
/*
 * PhysicalComponent.java
 *
 * Created on January 6, 2004, 2:13 PM
 */

package CCSDT;
import java.util.*;
import javax.swing.*;
```

```
import javax.swing.JOptionPane;
import javax.swing.JList;
import java.io.File;
import java.io.*;

/**
 *
 * @author  kjohns
 *This class is the graphical representation of a CatalogComponent descriptor.
 *This is where the CatalogComponent form is managed and the CatalogComponent
 *toString() resides.
 */
public class PhysicalComponent extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form PhysicalComponent */
    public PhysicalComponent()
    {
        initComponents();
        IsTemplate = false;
        mProceduralSettingFileName = new String();
        networkConnectionFinalFileListVec = new Vector();
        networkConnectionFinalElementListVec = new Vector();
        mSoftware_Vector = new Vector();
        mRemoteUserVector = new Vector();
        mLocalUserVector = new Vector();;
        mAuthenticationVector = new Vector();
        mAssetVector = new Vector();
        software_added =  false;
        assets_added = false;
        trustedHost_added = false;
        auth_server_added = false;
        filter_added = false;
        remoteUsers_added = false;
        localUsers_added = false;
        network_added = false;
        procedure_added = false;
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        configMgmntbuttonGroup = new javax.swing.ButtonGroup();
        patchFreqbuttonGroup = new javax.swing.ButtonGroup();
        antiVirusbuttonGroup = new javax.swing.ButtonGroup();
        browserbuttonGroup = new javax.swing.ButtonGroup();
        emailbuttonGroup = new javax.swing.ButtonGroup();
        physcompScrollPane = new javax.swing.JScrollPane();
        physcompPanel = new javax.swing.JPanel();
        comboBoxandTextPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        descriptionScrollPane = new javax.swing.JScrollPane();
        descriptionTextArea = new javax.swing.JTextArea();
        costTextField = new javax.swing.JTextField();
        resaleTextField = new javax.swing.JTextField();
        maintenanceTextField = new javax.swing.JTextField();
        availabilityTextField = new javax.swing.JTextField();
        availablityLabel = new javax.swing.JLabel();
        maintenanceLabel = new javax.swing.JLabel();
        resaleLabel = new javax.swing.JLabel();
        costLabel = new javax.swing.JLabel();
```

```
descriptionLabel = new javax.swing.JLabel();
listsPanel = new javax.swing.JPanel();
softwareLabel = new javax.swing.JLabel();
softwareSourceScrollPane = new javax.swing.JScrollPane();
softwareSourceList = new javax.swing.JList();
softwareMoveButton = new javax.swing.JButton();
softwareFinalScrollPane = new javax.swing.JScrollPane();
softwareFinalList = new javax.swing.JList();
softwareRemoveButton = new javax.swing.JButton();
authServListLabel = new javax.swing.JLabel();
authSourceScrollPane = new javax.swing.JScrollPane();
authSourceList = new javax.swing.JList();
authMoveButton = new javax.swing.JButton();
authFinalScrollPane = new javax.swing.JScrollPane();
authFinalList = new javax.swing.JList();
authRemoveButton = new javax.swing.JButton();
networkLabel = new javax.swing.JLabel();
networkSourceScrollPane = new javax.swing.JScrollPane();
networkSourceList = new javax.swing.JList();
networkMoveButton = new javax.swing.JButton();
networkFinalScrollPane = new javax.swing.JScrollPane();
networkFinalList = new javax.swing.JList();
networkRemoveButton = new javax.swing.JButton();
assetsLabel = new javax.swing.JLabel();
assetsSourceScrollPane = new javax.swing.JScrollPane();
assetsSourceList = new javax.swing.JList();
assetsMoveButton = new javax.swing.JButton();
assetsFinalScrollPane = new javax.swing.JScrollPane();
assetsFinalList = new javax.swing.JList();
assetsRemoveButton = new javax.swing.JButton();
localUserLabel = new javax.swing.JLabel();
localUserSourceScrollPane = new javax.swing.JScrollPane();
localUserSourceList = new javax.swing.JList();
localUserMoveButton = new javax.swing.JButton();
localUserFinalScrollPane = new javax.swing.JScrollPane();
localUserFinalList = new javax.swing.JList();
localUserRemoveButton = new javax.swing.JButton();
remoteUserLabel = new javax.swing.JLabel();
remoteUserSourceScrollPane = new javax.swing.JScrollPane();
remoteUserSourceList = new javax.swing.JList();
remoteUserMoveButton = new javax.swing.JButton();
remoteUserFinalScrollPane = new javax.swing.JScrollPane();
remoteUserFinalList = new javax.swing.JList();
remoteUserRemoveButton = new javax.swing.JButton();
manyToOnePanel = new javax.swing.JPanel();
posIndexLabel = new javax.swing.JLabel();
zoneLabel = new javax.swing.JLabel();
userLabel = new javax.swing.JLabel();
procSettingsLabel = new javax.swing.JLabel();
hardwareNameLabel = new javax.swing.JLabel();
opSysLabel = new javax.swing.JLabel();
opSysSourceComboBox = new javax.swing.JComboBox();
hardwareSourceComboBox = new javax.swing.JComboBox();
procSettingsComboBox = new javax.swing.JComboBox();
userSourceComboBox = new javax.swing.JComboBox();
zonesourceComboBox = new javax.swing.JComboBox();
posIndexTextField = new javax.swing.JTextField();
zoneFinalTextField = new javax.swing.JTextField();
userFinalTextField = new javax.swing.JTextField();
procSettingsFinalTextField = new javax.swing.JTextField();
posIndexFinalComboBox = new javax.swing.JComboBox();
boolPanel = new javax.swing.JPanel();
```

```
cmPanel = new javax.swing.JPanel();
cMWeakRadioButton = new javax.swing.JRadioButton();
cMModerateRadioButton = new javax.swing.JRadioButton();
cMStrictRadioButton = new javax.swing.JRadioButton();
patchesPanel = new javax.swing.JPanel();
patchFreqAsReleasedRadioButton = new javax.swing.JRadioButton();
patchFreqRoutinelyRadioButton = new javax.swing.JRadioButton();
patchFreqAutoRadioButton = new javax.swing.JRadioButton();
avPanel = new javax.swing.JPanel();
antiVirusRegRadioButton = new javax.swing.JRadioButton();
antiVirusAutoRadioButton = new javax.swing.JRadioButton();
browserPanel = new javax.swing.JPanel();
browserLooseRadioButton = new javax.swing.JRadioButton();
browserNormalRadioButton = new javax.swing.JRadioButton();
browserStrictRadioButton = new javax.swing.JRadioButton();
emailPanel = new javax.swing.JPanel();
emailLooseRadioButton = new javax.swing.JRadioButton();
emailNormalRadioButton = new javax.swing.JRadioButton();
emailStrictRadioButton = new javax.swing.JRadioButton();
checkboxPanel = new javax.swing.JPanel();
assetProtectionCheckBox = new javax.swing.JCheckBox();
remoteAccessCheckBox = new javax.swing.JCheckBox();
pKICertsCheckBox = new javax.swing.JCheckBox();
oneTimePassCheckBox = new javax.swing.JCheckBox();
biometricsCheckBox = new javax.swing.JCheckBox();
tokenPKICheckBox = new javax.swing.JCheckBox();
clientPKICheckBox = new javax.swing.JCheckBox();
vPNClientCheckBox = new javax.swing.JCheckBox();
scanEmailCheckBox = new javax.swing.JCheckBox();
stripEmailCheckBox = new javax.swing.JCheckBox();
autoLogoutCheckBox = new javax.swing.JCheckBox();
userAdminNoMACCheckBox = new javax.swing.JCheckBox();
userAdminMACCheckBox = new javax.swing.JCheckBox();
adminOnlyInstallsCheckBox = new javax.swing.JCheckBox();
blockMediaCheckBox = new javax.swing.JCheckBox();
blockWriteCheckBox = new javax.swing.JCheckBox();
uPSCheckBox = new javax.swing.JCheckBox();
regularBackupCheckBox = new javax.swing.JCheckBox();
offSiteBackupsCheckBox = new javax.swing.JCheckBox();

setLayout(new java.awt.GridLayout(1, 0));

physcompScrollPane.setPreferredSize(new java.awt.Dimension(674, 2983));
physcompPanel.setLayout(new java.awt.GridBagLayout());

physcompPanel.setMinimumSize(new java.awt.Dimension(400, 340));
physcompPanel.setPreferredSize(new java.awt.Dimension(670, 2108));
comboBoxandTextPanel.setLayout(new java.awt.GridBagLayout());

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Component Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
comboBoxandTextPanel.add(nameTextField, gridBagConstraints);

descriptionScrollPane.setPreferredSize(new java.awt.Dimension(425, 60));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setWrapStyleWord(true);
descriptionTextArea.setMinimumSize(new java.awt.Dimension(0, 0));
descriptionTextArea.setPreferredSize(new java.awt.Dimension(425, 1000));
descriptionScrollPane.setViewportView(descriptionTextArea);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel.add(descriptionScrollPane, gridBagConstraints);

costTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel.add(costTextField, gridBagConstraints);

resaleTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel.add(resaleTextField, gridBagConstraints);

maintenanceTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel.add(maintenanceTextField, gridBagConstraints);

availabilityTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
comboBoxandTextPanel.add(availabilityTextField, gridBagConstraints);

availablityLabel.setLabelFor(availabilityTextField);
availablityLabel.setText("Percentage of Time Available:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel.add(availablityLabel, gridBagConstraints);

maintenanceLabel.setLabelFor(maintenanceTextField);
maintenanceLabel.setText("Maintenance Cost:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel.add(maintenanceLabel, gridBagConstraints);

resaleLabel.setLabelFor(resaleTextField);
resaleLabel.setText("Resale Value:");
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel.add(resaleLabel, gridBagConstraints);

costLabel.setLabelFor(costTextField);
costLabel.setText("Purchase Price:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
comboBoxandTextPanel.add(costLabel, gridBagConstraints);

descriptionLabel.setLabelFor(descriptionTextArea);
descriptionLabel.setText("Component Description:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
gridBagConstraints.insets = new java.awt.Insets(4, 0, 0, 0);
comboBoxandTextPanel.add(descriptionLabel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
physcompPanel.add(comboBoxandTextPanel, gridBagConstraints);

listsPanel.setLayout(new java.awt.GridBagLayout());

softwareLabel.setText("Software for this Device:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(softwareLabel, gridBagConstraints);

softwareSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
softwareSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
softwareSourceScrollPane.setViewportView(softwareSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(softwareSourceScrollPane, gridBagConstraints);

softwareMoveButton.setText(">>>>>");
softwareMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      softwareMoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
listsPanel.add(softwareMoveButton, gridBagConstraints);

softwareFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
softwareFinalScrollPane.setViewportView(softwareFinalList);
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 1;
listsPanel.add(softwareFinalScrollPane, gridBagConstraints);

softwareRemoveButton.setText("Remove");
softwareRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      softwareRemoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 2;
listsPanel.add(softwareRemoveButton, gridBagConstraints);

authServListLabel.setText("Authentication Server List:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(authServListLabel, gridBagConstraints);

authSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
authSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
authSourceScrollPane.setViewportView(authSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(authSourceScrollPane, gridBagConstraints);

authMoveButton.setText(">>>>>");
authMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      authMoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
listsPanel.add(authMoveButton, gridBagConstraints);

authFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
authFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
authFinalScrollPane.setViewportView(authFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
listsPanel.add(authFinalScrollPane, gridBagConstraints);

authRemoveButton.setText("Remove");
authRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      authRemoveButtonActionPerformed(evt);
   }
});
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
listsPanel.add(authRemoveButton, gridBagConstraints);

networkLabel.setText("Network Connections:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(networkLabel, gridBagConstraints);

networkSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
networkSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
networkSourceScrollPane.setViewportView(networkSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(networkSourceScrollPane, gridBagConstraints);

networkMoveButton.setText(">>>>>");
networkMoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      networkMoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 5;
listsPanel.add(networkMoveButton, gridBagConstraints);

networkFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
networkFinalScrollPane.setViewportView(networkFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 5;
listsPanel.add(networkFinalScrollPane, gridBagConstraints);

networkRemoveButton.setText("Remove");
networkRemoveButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      networkRemoveButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 6;
listsPanel.add(networkRemoveButton, gridBagConstraints);

assetsLabel.setText("Assets:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(assetsLabel, gridBagConstraints);
```

```
assetsSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
assetsSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
assetsSourceScrollPane.setViewportView(assetsSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 9;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(assetsSourceScrollPane, gridBagConstraints);

assetsMoveButton.setText(">>>>>");
assetsMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetsMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 9;
listsPanel.add(assetsMoveButton, gridBagConstraints);

assetsFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
assetsFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
assetsFinalScrollPane.setViewportView(assetsFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 9;
listsPanel.add(assetsFinalScrollPane, gridBagConstraints);

assetsRemoveButton.setText("Remove");
assetsRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetsRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 10;
listsPanel.add(assetsRemoveButton, gridBagConstraints);

localUserLabel.setText("Local User Access List:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(localUserLabel, gridBagConstraints);

localUserSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
localUserSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
localUserSourceScrollPane.setViewportView(localUserSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 11;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(localUserSourceScrollPane, gridBagConstraints);

localUserMoveButton.setText(">>>>>");
localUserMoveButton.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        localUserMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 11;
listsPanel.add(localUserMoveButton, gridBagConstraints);

localUserFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
localUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
localUserFinalScrollPane.setViewportView(localUserFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 11;
listsPanel.add(localUserFinalScrollPane, gridBagConstraints);

localUserRemoveButton.setText("Remove");
localUserRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        localUserRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 12;
listsPanel.add(localUserRemoveButton, gridBagConstraints);

remoteUserLabel.setText("Remote User Access List:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
listsPanel.add(remoteUserLabel, gridBagConstraints);

remoteUserSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
remoteUserSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
remoteUserSourceScrollPane.setViewportView(remoteUserSourceList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 13;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
listsPanel.add(remoteUserSourceScrollPane, gridBagConstraints);

remoteUserMoveButton.setText(">>>>>");
remoteUserMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        remoteUserMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 13;
listsPanel.add(remoteUserMoveButton, gridBagConstraints);

remoteUserFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
remoteUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
```

```
remoteUserFinalScrollPane.setViewportView(remoteUserFinalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 13;
listsPanel.add(remoteUserFinalScrollPane, gridBagConstraints);

remoteUserRemoveButton.setText("Remove");
remoteUserRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        remoteUserRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 14;
listsPanel.add(remoteUserRemoveButton, gridBagConstraints);

manyToOnePanel.setLayout(new java.awt.GridBagLayout());

posIndexLabel.setText("Grid Position in Office:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(posIndexLabel, gridBagConstraints);

zoneLabel.setText("Zone:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(zoneLabel, gridBagConstraints);

userLabel.setText("Assigned User:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(userLabel, gridBagConstraints);

procSettingsLabel.setText("Procedural Settings:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(procSettingsLabel, gridBagConstraints);

hardwareNameLabel.setText("Base Component for this Device:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(hardwareNameLabel, gridBagConstraints);

opSysLabel.setText("Operating System:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
manyToOnePanel.add(opSysLabel, gridBagConstraints);
```

```
opSysSourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
manyToOnePanel.add(opSysSourceComboBox, gridBagConstraints);

hardwareSourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
manyToOnePanel.add(hardwareSourceComboBox, gridBagConstraints);

procSettingsComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
procSettingsComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        procSettingsComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
manyToOnePanel.add(procSettingsComboBox, gridBagConstraints);

userSourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
userSourceComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        userSourceComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
manyToOnePanel.add(userSourceComboBox, gridBagConstraints);

zonesourceComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
zonesourceComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        zonesourceComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
manyToOnePanel.add(zonesourceComboBox, gridBagConstraints);

posIndexTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 0;
manyToOnePanel.add(posIndexTextField, gridBagConstraints);

zoneFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
manyToOnePanel.add(zoneFinalTextField, gridBagConstraints);

userFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
manyToOnePanel.add(userFinalTextField, gridBagConstraints);

procSettingsFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
manyToOnePanel.add(procSettingsFinalTextField, gridBagConstraints);

posIndexFinalComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
posIndexFinalComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        posIndexFinalComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
manyToOnePanel.add(posIndexFinalComboBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
listsPanel.add(manyToOnePanel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
physcompPanel.add(listsPanel, gridBagConstraints);

boolPanel.setLayout(new java.awt.GridBagLayout());

boolPanel.setBorder(new javax.swing.border.TitledBorder("Configuration Settings"));
cmPanel.setLayout(new java.awt.GridBagLayout());

cmPanel.setBorder(new javax.swing.border.TitledBorder("Configuration Managment Settings"));
cmPanel.setPreferredSize(new java.awt.Dimension(200, 97));
cMWeakRadioButton.setText("Weak");
configMgmntbuttonGroup.add(cMWeakRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
cmPanel.add(cMWeakRadioButton, gridBagConstraints);

cMModerateRadioButton.setText("Moderate");
configMgmntbuttonGroup.add(cMModerateRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
cmPanel.add(cMModerateRadioButton, gridBagConstraints);

cMStrictRadioButton.setSelected(true);
cM StrictRadioButton.setText("Strict");
configMgmntbuttonGroup.add(cMStrictRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
```

```
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
cmPanel.add(cMStrictRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
boolPanel.add(cmPanel, gridBagConstraints);

patchesPanel.setLayout(new java.awt.GridBagLayout());

patchesPanel.setBorder(new javax.swing.border.TitledBorder("Patch Update Frequency"));
patchesPanel.setPreferredSize(new java.awt.Dimension(200, 97));
patchFreqAsReleasedRadioButton.setText("As Released");
patchFreqbuttonGroup.add(patchFreqAsReleasedRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqAsReleasedRadioButton, gridBagConstraints);

patchFreqRoutinelyRadioButton.setText("Routinely");
patchFreqbuttonGroup.add(patchFreqRoutinelyRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqRoutinelyRadioButton, gridBagConstraints);

patchFreqAutoRadioButton.setSelected(true);
patchFreqAutoRadioButton.setText("Automatically");
patchFreqbuttonGroup.add(patchFreqAutoRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
patchesPanel.add(patchFreqAutoRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
boolPanel.add(patchesPanel, gridBagConstraints);

avPanel.setLayout(new java.awt.GridBagLayout());

avPanel.setBorder(new javax.swing.border.TitledBorder("Antivirus Update Frequency"));
avPanel.setPreferredSize(new java.awt.Dimension(200, 73));
antiVirusRegRadioButton.setText("Regular");
antiVirusbuttonGroup.add(antiVirusRegRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
avPanel.add(antiVirusRegRadioButton, gridBagConstraints);

antiVirusAutoRadioButton.setSelected(true);
antiVirusAutoRadioButton.setText("Automatically");
antiVirusbuttonGroup.add(antiVirusAutoRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
avPanel.add(antiVirusAutoRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
boolPanel.add(avPanel, gridBagConstraints);

browserPanel.setLayout(new java.awt.GridBagLayout());

browserPanel.setBorder(new javax.swing.border.TitledBorder("Browser Settings"));
browserPanel.setPreferredSize(new java.awt.Dimension(200, 103));
browserLooseRadioButton.setText("Loose");
browserbuttonGroup.add(browserLooseRadioButton);
browserLooseRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserLooseRadioButton, gridBagConstraints);

browserNormalRadioButton.setText("Normal");
browserbuttonGroup.add(browserNormalRadioButton);
browserNormalRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserNormalRadioButton, gridBagConstraints);

browserStrictRadioButton.setSelected(true);
browserStrictRadioButton.setText("Strict");
browserbuttonGroup.add(browserStrictRadioButton);
browserStrictRadioButton.setBorder(new javax.swing.border.TitledBorder(""));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
browserPanel.add(browserStrictRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
boolPanel.add(browserPanel, gridBagConstraints);

emailPanel.setLayout(new java.awt.GridBagLayout());

emailPanel.setBorder(new javax.swing.border.TitledBorder("Email Settings"));
emailPanel.setPreferredSize(new java.awt.Dimension(200, 97));
emailLooseRadioButton.setText("Loose");
emailbuttonGroup.add(emailLooseRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
emailPanel.add(emailLooseRadioButton, gridBagConstraints);

emailNormalRadioButton.setText("Normal");
emailbuttonGroup.add(emailNormalRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
emailPanel.add(emailNormalRadioButton, gridBagConstraints);

emailStrictRadioButton.setSelected(true);
emailStrictRadioButton.setText("Strict");
emailbuttonGroup.add(emailStrictRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 18;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
emailPanel.add(emailStrictRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
boolPanel.add(emailPanel, gridBagConstraints);

checkboxPanel.setLayout(new java.awt.GridBagLayout());

assetProtectionCheckBox.setText("Access controls fully and correctly confgured at startup");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(assetProtectionCheckBox, gridBagConstraints);

remoteAccessCheckBox.setText("Remote access requires authentication");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(remoteAccessCheckBox, gridBagConstraints);

pKICertsCheckBox.setText("Accept PKI Certs");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(pKICertsCheckBox, gridBagConstraints);

oneTimePassCheckBox.setText("Use one time password token");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(oneTimePassCheckBox, gridBagConstraints);

biometricsCheckBox.setText("Use biometrics");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(biometricsCheckBox, gridBagConstraints);

tokenPKICheckBox.setText("Use token PKI certs");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(tokenPKICheckBox, gridBagConstraints);

clientPKICheckBox.setText("Use client PKI certs");
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(clientPKICheckBox, gridBagConstraints);

vPNClientCheckBox.setText("VPN client");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(vPNClientCheckBox, gridBagConstraints);

scanEmailCheckBox.setText("Scan email attachements");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 10;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(scanEmailCheckBox, gridBagConstraints);

stripEmailCheckBox.setText("Strip email attachments");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(stripEmailCheckBox, gridBagConstraints);

autoLogoutCheckBox.setText("Automatic lock/logout");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(autoLogoutCheckBox, gridBagConstraints);

userAdminNoMACCheckBox.setText("User can administrate non-MAC security attributres");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(userAdminNoMACCheckBox, gridBagConstraints);

userAdminMACCheckBox.setText("User can administrate MAC security attributres");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(userAdminMACCheckBox, gridBagConstraints);

adminOnlyInstallsCheckBox.setText("Only administrator can install software");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 15;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(adminOnlyInstallsCheckBox, gridBagConstraints);

blockMediaCheckBox.setText("Block use of removable media and ports");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(blockMediaCheckBox, gridBagConstraints);
```

```
        blockWriteCheckBox.setText("Block write access to local storage");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 17;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(blockWriteCheckBox, gridBagConstraints);

        uPSCheckBox.setText("Component is connected to a UPS");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 18;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(uPSCheckBox, gridBagConstraints);

        regularBackupCheckBox.setText("Administrators routinely backup this component");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(regularBackupCheckBox, gridBagConstraints);

        offSiteBackupsCheckBox.setText("Backups are stored offsite");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        checkboxPanel.add(offSiteBackupsCheckBox, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
        boolPanel.add(checkboxPanel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        physcompPanel.add(boolPanel, gridBagConstraints);

        physcompScrollPane.setViewportView(physcompPanel);

        add(physcompScrollPane);

    }
//the following listeners set a text field based on a combo box selection
    private void posIndexFinalComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(posIndexFinalComboBox.getSelectedItem() != null)
        {
            posIndexTextField.setText(
                  String.valueOf((posIndexFinalComboBox.getSelectedIndex())-1));
        }
    }

    private void procSettingsComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(procSettingsComboBox.getSelectedIndex() >= 0)
        {
            procSettingsFinalTextField.setText(
                proceduralSettingsElementListVec.get(
                        procSettingsComboBox.getSelectedIndex()).
                            toString());
```

```
            mProceduralSettingFileName = proceduralSettingsFileListVec.get(
                  procSettingsComboBox.getSelectedIndex()).toString();
            procedure_added = true;
         }
      }

      private void userSourceComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
         if(userSourceComboBox.getSelectedItem() != null)
         {
            userFinalTextField.setText(
                        userSourceComboBox.getSelectedItem().toString());
         }
      }

      private void zonesourceComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
         if(zonesourceComboBox.getSelectedItem() != null)
         {
            zoneFinalTextField.setText(
                        zonesourceComboBox.getSelectedItem().toString());
         }
      }
   //the following listeners are for the buttons that either move or remove
      //entries from the acl and cost list, list boxes
      private void remoteUserRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
      {
         if(remoteUserFinalList.getSelectedIndex() >= 0)
         {
            mRemoteUserVector.remove(remoteUserFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)remoteUserFinalList.getModel();
            listModel.remove(remoteUserFinalList.getSelectedIndex());
            remoteUserFinalList = new JList(listModel);
            remoteUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            remoteUserFinalScrollPane.setViewportView(remoteUserFinalList);
         }
      }

      private void remoteUserMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
      {
         if(remoteUserSourceList.getSelectedValue() != null)
         {
            mRemoteUserVector.addElement(remoteUserSourceList.getSelectedValue().toString());
            Object tempObject = new Object();
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < mRemoteUserVector.size(); i++)
            {
               tempObject = mRemoteUserVector.get(i);
               listModel.addElement(tempObject);
            }
            remoteUserFinalList = null;
            remoteUserFinalList = new JList(listModel);
            remoteUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            remoteUserFinalScrollPane.setViewportView(remoteUserFinalList);
            remoteUsers_added = true;
         }
      }

      private void localUserRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
      {
         if(localUserFinalList.getSelectedIndex() >= 0)
         {
            mLocalUserVector.remove(localUserFinalList.getSelectedIndex());
```

```
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)localUserFinalList.getModel();
            listModel.remove(localUserFinalList.getSelectedIndex());
            localUserFinalList = new JList(listModel);
            localUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            localUserFinalScrollPane.setViewportView(localUserFinalList);
        }
    }

    private void localUserMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(localUserSourceList.getSelectedValue() != null)
        {
            mLocalUserVector.addElement(localUserSourceList.getSelectedValue().toString());
            Object tempObject = new Object();
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < mLocalUserVector.size(); i++)
            {
                tempObject = mLocalUserVector.get(i);
                listModel.addElement(tempObject);
            }
            localUserFinalList = null;
            localUserFinalList = new JList(listModel);
            localUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            localUserFinalScrollPane.setViewportView(localUserFinalList);
            localUsers_added = true;
        }
    }

    private void networkRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(networkFinalList.getSelectedIndex() >= 0)
        {
            networkConnectionFinalElementListVec.remove(networkFinalList.getSelectedIndex());
            networkConnectionFinalFileListVec.remove(networkFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)networkFinalList.getModel();
            listModel.remove(networkFinalList.getSelectedIndex());
            networkFinalList = new JList(listModel);
            networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            networkFinalScrollPane.setViewportView(networkFinalList);
        }
    }

    private void networkMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(networkSourceList.getSelectedValue() != null)
        {
            Object tempObject = new Object();
            //add the element name to the final element vector

networkConnectionFinalElementListVec.addElement(networkSourceList.getSelectedValue().toString());
            //based on the index of the source use it to copy the set file name to the final file name vector

networkConnectionFinalFileListVec.addElement(networkConnectionSourceFileListVec.get(networkSourceList.getSel
ectedIndex()).toString());
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < networkConnectionFinalElementListVec.size(); i++)
            {
                tempObject = networkConnectionFinalElementListVec.get(i);
                listModel.addElement(tempObject);
            }
```

```
        networkFinalList = null;
        networkFinalList = new JList(listModel);
        networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        networkFinalScrollPane.setViewportView(networkFinalList);
        network_added = true;
    }
}

private void authRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(authFinalList.getSelectedIndex() >= 0)
    {
        mAuthenticationVector.remove(authFinalList.getSelectedIndex());
        DefaultListModel listModel = new DefaultListModel();
        listModel = (DefaultListModel)authFinalList.getModel();
        listModel.remove(authFinalList.getSelectedIndex());
        authFinalList = new JList(listModel);
        authFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        authFinalScrollPane.setViewportView(authFinalList);
    }
}

private void authMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(authSourceList.getSelectedValue() != null)
    {
        mAuthenticationVector.addElement(authSourceList.getSelectedValue().toString());
        Object tempObject = new Object();
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < mAuthenticationVector.size(); i++)
        {
            tempObject = mAuthenticationVector.get(i);
            listModel.addElement(tempObject);
        }
        authFinalList = null;
        authFinalList = new JList(listModel);
        authFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        authFinalScrollPane.setViewportView(authFinalList);
        auth_server_added = true;
    }
}

private void softwareRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(softwareFinalList.getSelectedIndex() >= 0)
    {
        mSoftware_Vector.remove(softwareFinalList.getSelectedIndex());
        DefaultListModel listModel = new DefaultListModel();
        listModel = (DefaultListModel)softwareFinalList.getModel();
        listModel.remove(softwareFinalList.getSelectedIndex());
        softwareFinalList = new JList(listModel);
        softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        softwareFinalScrollPane.setViewportView(softwareFinalList);
    }
}

private void softwareMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    if(softwareSourceList.getSelectedValue() != null)
    {
        Object tempObject = new Object();
        mSoftware_Vector.addElement(softwareSourceList.getSelectedValue().toString());
```

```
      DefaultListModel listModel = new DefaultListModel();
      for(int i = 0; i < mSoftware_Vector.size(); i++)
      {
         tempObject = mSoftware_Vector.get(i);
         listModel.addElement(tempObject);
      }
      softwareFinalList = null;
      softwareFinalList = new JList(listModel);
      softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
      softwareFinalScrollPane.setViewportView(softwareFinalList);
      software_added = true;
   }
}

private void assetsRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
   if(assetsFinalList.getSelectedIndex() >= 0)
   {
      mAssetVector.remove(assetsFinalList.getSelectedIndex());
      DefaultListModel listModel = new DefaultListModel();
      listModel = (DefaultListModel)assetsFinalList.getModel();
      listModel.remove(assetsFinalList.getSelectedIndex());
      assetsFinalList = new JList(listModel);
      assetsFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
      assetsFinalScrollPane.setViewportView(assetsFinalList);
   }
}

private void assetsMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
{
   if(assetsSourceList.getSelectedValue() != null)
   {
      Object tempObject = new Object();
      mAssetVector.addElement(assetsSourceList.getSelectedValue().toString());
      DefaultListModel listModel = new DefaultListModel();
      for(int i = 0; i < mAssetVector.size(); i++)
      {
         tempObject = mAssetVector.get(i);
         listModel.addElement(tempObject);
      }
      assetsFinalList = null;
      assetsFinalList = new JList(listModel);
      assetsFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
      assetsFinalScrollPane.setViewportView(assetsFinalList);
      assets_added = true;
   }
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
```

```
//element in the vector add it to the combo box
     // if the target of the vector is a list box just add the
     // vector directly
public void populateStaticSourceLists(java.util.Vector aBase,
       java.util.Vector aOpSys, java.util.Vector aPassComp,
      java.util.Vector aPassLen, java.util.Vector aSoftware)
{
   //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
   //to populate:
   //base components
   //OS
   //password complexity
   //password length
   //software

   //base component
   for(int i = 0; i < aBase.size(); i++)
   {
      hardwareSourceComboBox.addItem(aBase.get(i));
   }
   //OS
   for(int i = 0; i < aOpSys.size(); i++)
   {
      opSysSourceComboBox.addItem(aOpSys.get(i));
   }
   //software
   softwareSourceList.setListData(aSoftware);
}

//populateDynamicSourceLists() takes datapath and startupScenario as
//parameters in ScenarioDefinitionTool.java and is used to add dynamic
//content to combo boxes and list boxes.
//First the scenairo is used to get the added reusable sets
//Second any combo boxes to be used are cleared and reinitialized
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
{
   //uses the Scenario passed to it to set the dynamic values of
   //dependent sets
   //Assets+
   //Authorization Server List
   //Local User List
   //NetworkConnection+
   //Procedural Settings+
   //workspace = position index+
   //Remote User List
   //trused hosts
   //User+
   //Zone+
   //DACGroup
   mScenario = aScenario;
   Object tempObj = new Object();
   Vector tempVec = new Vector();
   ScenarioElementSet tempSet = new ScenarioElementSet();
   assetListVec = new Vector();
   authServerListVec = new Vector();
   userListVec = new Vector();
   zoneListVec = new Vector();
   proceduralSettingsFileListVec = new Vector();
   proceduralSettingsElementListVec = new Vector();
```

```
                networkConnectionSourceFileListVec = new Vector();
                networkConnectionSourceElementListVec = new Vector();
                assetsSourceList.removeAll();
                authSourceList.removeAll();
                networkSourceList.removeAll();
                //Assets
                tempObj = aScenario.scenarioManager.get(0);
                tempVec = (Vector)tempObj;
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Asset/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   for(int j = 0; j < tempSet.elementContainer.size(); j++)
                   {
                      Asset tempAsset = new Asset();
                      tempObj = tempSet.elementContainer.get(j);
                      tempAsset = (Asset)tempObj;
                      assetListVec.add(tempAsset.getNameTextField());
                   }
                }
                assetsSourceList.setListData(assetListVec);
                tempObj = tempVec = null;
                //Authentication Server List
                tempObj = aScenario.scenarioManager.get(8);
                tempVec = (Vector)tempObj;
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
```

```
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Physical
Component/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                   "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       PhysicalComponent tempPhysComp = new PhysicalComponent();
                       tempObj = tempSet.elementContainer.get(j);
                       tempPhysComp = (PhysicalComponent)tempObj;
                       authServerListVec.add(tempPhysComp.getNameTextField());
                    }
                 }
                 authSourceList.setListData(authServerListVec);
                 tempObj = tempVec = null;
                 //Local User List + remote user + user
                 tempObj = aScenario.scenarioManager.get(11);
                 tempVec = (Vector)tempObj;
                 userSourceComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
```

```
                        JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                   JOptionPane.showMessageDialog(this,
                        "Exception during file read - the object was not found",
                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file read",
                              "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   input.close();
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file close",
                                 "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                   User tempUser = new User();
                   tempObj = tempSet.elementContainer.get(j);
                   tempUser = (User)tempObj;
                   userListVec.add(tempUser.getNameTextField());
                   userSourceComboBox.addItem(tempUser.getNameTextField());
                }
             }
             localUserSourceList.setListData(userListVec);
             remoteUserSourceList.setListData(userListVec);
             tempObj = tempVec = null;
             //DACGroup
             //all of the DACGroups in the scenario need to appear in the local
             //and remote user access lists.  The IO below will get the DACGroup
             //and add it to the already populated user list.  The end result
             //will be that all the users appear followed by all of the
             //DACGroups.
             tempObj = aScenario.scenarioManager.get(4);
             tempVec = (Vector)tempObj;
             for(int i = 0; i < tempVec.size(); i++)
             {
                tempObj = tempVec.get(i);
                try
                {
                   input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this,
                        "Exception during input stream creation", "Exception",
                        JOptionPane.ERROR_MESSAGE);
                }
                try
                {
```

```
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                    DACGroup tempDACGroup = new DACGroup();
                    tempObj = tempSet.elementContainer.get(j);
                    tempDACGroup = (DACGroup)tempObj;
                    userListVec.add(tempDACGroup.getNameTextField());
                }
            }
            localUserSourceList.setListData(userListVec);
            remoteUserSourceList.setListData(userListVec);
            tempObj = tempVec = null;
            //NetworkConnection
            tempObj = aScenario.scenarioManager.get(14);
            tempVec = (Vector)tempObj;
            tempObj = tempVec.get(2);//get the networkconnection
            tempVec = (Vector)tempObj;
            for(int i = 0; i < tempVec.size(); i++)
            {
                tempObj = tempVec.get(i);
                try
                {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Other/Component Network
Connection/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
```

```
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       //save the file names
                       networkConnectionSourceFileListVec.addElement(tempSet.getelementSetName());
                       tempObj = tempSet.elementContainer.get(j);
                       NetworkConnection tempNetworkConnection = new NetworkConnection();
                       tempNetworkConnection = (NetworkConnection)tempObj;
                       //save the element name - will be used in the list

networkConnectionSourceElementListVec.addElement(tempNetworkConnection.getNameTextField());
                    }
                 }
                 networkSourceList.setListData(networkConnectionSourceElementListVec);
                 tempObj = tempVec = null;
                 //Procedural Settings
                 tempObj = aScenario.scenarioManager.get(14);
                 tempVec = (Vector)tempObj;
                 tempObj = tempVec.get(1);//get the proc settings
                 tempVec = (Vector)tempObj;
                 procSettingsComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural
Settings/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
```

```
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                    //save the set filename for later
                    proceduralSettingsFileListVec.addElement(tempSet.getelementSetName());
                    tempObj = tempSet.elementContainer.get(j);
                    ProceduralSettings tempProcSettings = new ProceduralSettings();
                    tempProcSettings = (ProceduralSettings)tempObj;
                    //save the element name for later
                    proceduralSettingsElementListVec.addElement(tempProcSettings.getNameTextField());
                    procSettingsComboBox.addItem(tempProcSettings.getNameTextField());
                }
            }
            tempObj = tempVec = null;
            //workspace
            tempObj = aScenario.scenarioManager.get(12);
            tempVec = (java.util.Vector)tempObj;
            posIndexFinalComboBox.removeAllItems();
//          for(int i = 0; i < tempVec.size(); i++)
//          {
            if(tempVec.size() > 0)
            {
                tempObj = tempVec.get(0);
                try
                {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Workspace/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
```

```
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
//              for(int j = 0; j < tempSet.elementContainer.size(); j++)
//              {
                    Workspace tempWorkspace = new Workspace();
                    tempObj = tempSet.elementContainer.get(0);
                    tempWorkspace = (Workspace)tempObj;
                    DefaultListModel listModel = new DefaultListModel();
                    listModel = (DefaultListModel)tempWorkspace.getListModel();
                    for(int i = 0; i < listModel.size(); i++)
                    {
                        posIndexFinalComboBox.addItem(listModel.get(i).toString());
                    }
//              }
//          }
                    tempObj = tempVec = null;
                }
                //Zone
                tempObj = aScenario.scenarioManager.get(13);
                tempVec = (Vector)tempObj;
                zonesourceComboBox.removeAllItems();
                for(int i = 0; i < tempVec.size(); i++)
                {
                    tempObj = tempVec.get(i);
                    try
                    {
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Zone/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showM essageDialog(this,
                                "Exception during input stream creation", "Exception",
                                JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                        JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read",
                                        "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        input.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file close",
```

```
                                "Exception", JOptionPane.ERROR_MESSAGE);
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
           Zone tempZone = new Zone();
           tempObj = tempSet.elementContainer.get(j);
           tempZone = (Zone)tempObj;
           zoneListVec.add(tempZone.getNameTextField());
           zonesourceComboBox.addItem(tempZone.getNameTextField());
        }
     }
  }

  public void Load()
  {
  }

  public void Save()
  {
  }

  //After save operations cause the list boxes in forms to be cleared
  //this method repopulates those list boxes and reinitilizes them.
  //The code in this method is literally taken line for line from the
  //button listeners above.
  public void reInitLists(Vector aSoftware)
  {
     DefaultListModel listModel = new DefaultListModel();
     Object tempObject = new Object();

     //software lists
     softwareSourceList.setListData(aSoftware);
     for(int i = 0; i < mSoftware_Vector.size(); i++)
     {
        tempObject = mSoftware_Vector.get(i);
        listModel.addElement(tempObject);
     }
     softwareFinalList = null;
     softwareFinalList = new JList(listModel);
     softwareFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
     softwareFinalScrollPane.setViewportView(softwareFinalList);

     //authenitcation lists
     authSourceList.setListData(authServerListVec);
     listModel = listModel = new DefaultListModel();
     tempObject = new Object();
     for(int i = 0; i < mAuthenticationVector.size(); i++)
     {
        tempObject = mAuthenticationVector.get(i);
        listModel.addElement(tempObject);
     }
     authFinalList = null;
     authFinalList = new JList(listModel);
     authFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
     authFinalScrollPane.setViewportView(authFinalList);

     //network connection lists
     networkSourceList.setListData(networkConnectionSourceElementListVec);
     listModel = listModel = new DefaultListModel();
     tempObject = new Object();
     for(int i = 0; i < networkConnectionFinalElementListVec.size(); i++)
     {
```

```
            tempObject = networkConnectionFinalElementListVec.get(i);
            listModel.addElement(tempObject);
        }
        networkFinalList = null;
        networkFinalList = new JList(listModel);
        networkFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        networkFinalScrollPane.setViewportView(networkFinalList);

        //asset lists
        assetsSourceList.setListData(assetListVec);
        listModel = listModel = new DefaultListModel();
        tempObject = new Object();
        for(int i = 0; i < mAssetVector.size(); i++)
        {
            tempObject = mAssetVector.get(i);
            listModel.addElement(tempObject);
        }
        assetsFinalList = null;
        assetsFinalList = new JList(listModel);
        assetsFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        assetsFinalScrollPane.setViewportView(assetsFinalList);

        //local user lists
        localUserSourceList.setListData(userListVec);
        listModel = listModel = new DefaultListModel();
        tempObject = new Object();
        for(int i = 0; i < mLocalUserVector.size(); i++)
        {
            tempObject = mLocalUserVector.get(i);
            listModel.addElement(tempObject);
        }
        localUserFinalList = null;
        localUserFinalList = new JList(listModel);
        localUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        localUserFinalScrollPane.setViewportView(localUserFinalList);

        //remote user lists
        remoteUserSourceList.setListData(userListVec);
        listModel = listModel = new DefaultListModel();
        tempObject = new Object();
        for(int i = 0; i < mRemoteUserVector.size(); i++)
        {
            tempObject = mRemoteUserVector.get(i);
            listModel.addElement(tempObject);
        }
        remoteUserFinalList = null;
        remoteUserFinalList = new JList(listModel);
        remoteUserFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        remoteUserFinalScrollPane.setViewportView(remoteUserFinalList);
    }

    //This method provides a way to reinitialize the button listeners when
    //they die after IO operations.  The code is taken line for line from the
    //initComponents() method.
    public void reInitButtons()
    {
        softwareMoveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                softwareMoveButtonActionPerformed(evt);
            }
        });
```

```
softwareRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        softwareRemoveButtonActionPerformed(evt);
    }
});

authMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        authMoveButtonActionPerformed(evt);
    }
});

authRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        authRemoveButtonActionPerformed(evt);
    }
});

networkMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        networkMoveButtonActionPerformed(evt);
    }
});

networkRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        networkRemoveButtonActionPerformed(evt);
    }
});

assetsMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetsMoveButtonActionPerformed(evt);
    }
});

assetsRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetsRemoveButtonActionPerformed(evt);
    }
});

localUserMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        localUserMoveButtonActionPerformed(evt);
    }
});

localUserRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        localUserRemoveButtonActionPerformed(evt);
    }
});

remoteUserMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        remoteUserMoveButtonActionPerformed(evt);
    }
});

remoteUserRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
            remoteUserRemoveButtonActionPerformed(evt);
        }
    });
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario,
        java.util.Vector aBase,java.util.Vector aOpSys,
        java.util.Vector aPassComp, java.util.Vector aPassLen,
        java.util.Vector aSoftware)
{
    populateDynamicSourceLists(aPath, aScenario);
    populateStaticSourceLists(aBase, aOpSys, aPassComp, aPassLen,
    aSoftware);

    procSettingsComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            procSettingsComboBoxActionPerformed(evt);
        }
    });

    userSourceComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            userSourceComboBoxActionPerformed(evt);
        }
    });

    zonesourceComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            zonesourceComboBoxActionPerformed(evt);
        }
    });

    posIndexFinalComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            posIndexFinalComboBoxActionPerformed(evt);
        }
    });
}

//the following name_toString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller name_toString() methods are called from the primary
//asset toString() method.

//creates the software list formatted output
private String softwareList_toString()
{
    String softwareString = new String();
    DefaultListModel listModel = new DefaultListModel();
    if(software_added)
    {
        listModel = (DefaultListModel)softwareFinalList.getModel();
        softwareString +=
      "\t//List of software programs on this component from the enumerated"+
        " list of\n\t//software defined outside of this SDF.  If there are"+
        " not \"Software\" fields, then\n\t//the component has whatever "+
        "software is defined as part of the component\n\t//named in the"+
        "\"HW\" field.  If any fields are present, this component inherits"+
        "\n\t//none of the software from its base component.\n";
```

```
        for(int i = 0; i < listModel.getSize(); i++)
        {
           softwareString +=
           "\tSoftware: " +
           listModel.get(i).toString() +
           " :end\n";
        }
     }
     return softwareString;
}

//creates the asset list formatted output
private String assetList_toString()
{
     String assetString = new String();
     DefaultListModel listModel = new DefaultListModel();
     if(assets_added)
     {
        listModel = (DefaultListModel)assetsFinalList.getModel();
        assetString +=
        "\t//A list of Assets located on tihs machine\n";
        for(int i = 0; i < listModel.getSize(); i++)
        {
           assetString +=
           "\tAssets: " +
           listModel.get(i).toString() +
           " :end\n";
        }
     }
     return assetString;
}

//creates the local user access list formatted output
private String localUserList_toString()
{
     String localUserString = new String();
     DefaultListModel listModel = new DefaultListModel();
     if(localUsers_added)
     {
        listModel = (DefaultListModel)localUserFinalList.getModel();
        localUserString +=
        "\n\t//List of users who have explicit local access to this component."+
     " A workstation\n\t//having an \"AuthServerList\" entry permits"+
     " authentication of users defined in\n\t//those authentication servers"+
     " regardless of whether the user appears in the\n\t//workstaion's"+
     " \"AccessListLocal\"\n";
        for(int i = 0; i < listModel.getSize(); i++)
        {
           localUserString +=
           "\tAccessListLocal: " +
           listModel.get(i).toString() +
           " :end\n";
        }
     }
     return localUserString;
}

//creates the remote user access list formatted output
private String remoteUserList_toString()
{
     String remoteUserString = new String();
     DefaultListModel listModel = new DefaultListModel();
```

```
   if(remoteUsers_added)
   {
      listModel = (DefaultListModel)remoteUserFinalList.getModel();
      remoteUserString +=
      "\n\t//A list of users who have explicit remote access to this"+
   " component\n";
      for(int i = 0; i < listModel.getSize(); i++)
      {
         remoteUserString +=
         "\tAccessListRemote: " +
         listModel.get(i).toString() +
         " :end\n";
      }
   }
   return remoteUserString;
}

//creates the authentication list formatted output
private String authServerList_toString()
{
   String authServerString = new String();
   DefaultListModel listModel = new DefaultListModel();
   if(auth_server_added)
   {
      listModel = (DefaultListModel)authFinalList.getModel();
      authServerString +=
      "\n\t//kerberos or other type of of authentication servers used by"+
   " this component when\n\t//either authenticating to another component"+
   " or when some other component\n\t//is authenticating to this"+
   " component.  This names the actual servers\n\t//(by tag name)\n"+
   "\tAuthServerList:\n";
      for(int i = 0; i < listModel.getSize(); i++)
      {
         authServerString +=
         "\t\tName: " +
         listModel.get(i).toString() +
         " :end\n";
      }
      authServerString += "\t:end //AuthServerList\n\n";
   }
   return authServerString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an PhysicalComponent descriptor
public String toString(String aText)
{
   Object tempObj = new Object();
   Filter tempFilter = new Filter();
   NetworkConnection tempNetwork = new NetworkConnection();
   String outputString = new String();
   outputString =
   "//The start of the component section.\n"+
   "Component:\n\n"+
   "\t//The name of the component.  If \"IsTemplate\" is true, then this"+
   " is a catalogue\n\t//item that the player can purchase and \"Name\""+
   " should be a psuedo\n\t//brand name since it will show up in the"+
   " catalogue.\n\t//Catalogue items can also be referenced in\n"+
   "\t//subsequent Component definitions.  If \"IsTemplate\" is false,"+
   " then this\n\t//component is instantiated at the beginning of the "+
```

```
"game"+
" and \"Name\" should\n\t//identify the component within the context "+
"of"+
" the scenario, e.g.,\n\t//\"Bob's computer.\"\n"+
"\tName: "+
nameTextField.getText()+
" :end\n\n"+
"\t//If this boolean field is true, this component is a catalogue item"+
" that is\n\t//part of the buy screen.  If it is false, the component"+
" is instaniated at the start\n\t//of the game.\n"+
"\tIsTemplate: "+
String.valueOf(IsTemplate) +
" :end\n\n"+
"\t//A description of the component. If this is a catalogue item, this"+
" description will\n\t//appear in the ctalogue.  Otherwise, this"+
" description will be presented to the\n\t//player when the component"+
" itself is selected and it should be in the context\n\t//the"+
" scenario.\n"+
"\tDescription: "+
descriptionTextArea.getText() +
" :end\n\n"+
"\t//The AssetProtection is a boolean value that controls whether the"+
" intended\n\tassess controls (MAC and DAC) in the asset description"+
" is fully and\n\t//correctly configured on this machine at startup"+
".  If true then it will be correct,\n\t//if false then it will depend"+
"\n\t//on the skill ratings of the IT department.\n"+
"\tAssetProtection: "+
String.valueOf(assetProtectionCheckBox.isSelected()) +
" :end\n\n"+
"\t//The base component from which this component is derived.  This"+
"can\n\t//value can name either another defined \"Component\" from"+
"within this SDF, or\n\t//it can name one of the pre-defined "+
"components that are defined outside of the\n\t//SDF.\n"+
"\tHW: ";
if(hardwareSourceComboBox.getSelectedItem() != null)
{
    outputString += hardwareSourceComboBox.getSelectedItem().toString();
}
outputString += " :end\n\n"+
"\t//The dollar cost to purchase the component\n"+
"\tCost: "+
costTextField.getText() +
" :end\n\n"+
"\t//The dollar value of selling the component after it has been "+
"purchased\n"+
"\tResale: "+
resaleTextField.getText() +
" :end\n\n"+
"\t//The monthly maintenance cost of this component\n"+
"\tMaintenance: "+
maintenanceTextField.getText() +
" :end\n\n"+
"\t//The reliability of the component expressed as a percentage of the"+
" time that is usable\n"+
"\tAvailability: "+
availabilityTextField.getText() +
" :end\n\n"+
"\t//OS (if any) on this component.  Value is from the enumerated list"+
"of possible\n\t//operating systems.  If this value is not present, "+
"the component will have any\n\t//OS defined as part of the component"+
"named in the \"HW\" field.\n"+
"\tOS: ";
```

```
if(opSysSourceComboBox.getSelectedItem() != null)
{
   outputString += opSysSourceComboBox.getSelectedItem().toString();
}
outputString += " :end\n\n"+
softwareList_toString() +
"\n\t//Whether this component requires some form of authenication"+
" when being\n\t//accessed from a remote component.\n"+
"\tRemoteAuthentication: "+
String.valueOf(remoteAccessCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to accept PKI "+
" certificates for\n\t//authentication from remote components."+
"  The assurance depends\n\t//on the OS, this just determines"+
" if the certificates are used.\n"+
"\tAcceptPKICerts: "+
String.valueOf(pKICertsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to accept one-time"+
" passwords\n\t//from password-generation tokens.\n"+
"\tUseOneTimePasswordToken: "+
String.valueOf(oneTimePassCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use biometirc"+
" authentication\n\t//peripherals.\n"+
"\tUseBiometrics: "+
String.valueOf(biometricsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use physical"+
" tokens with\n\t//user-based client PKI certificates for"+
" authentication\n"+
"\tUseTokenPKICerts: "+
String.valueOf(tokenPKICheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use client PKI"+
" certificates managed\n\t//by the client for authenticating to"+
" remote components.  The assurance\n\t//depends on\n\t//the OS"+
", this just determines if the certificates are used.\n"+
"\tUseClientPKICerts: "+
String.valueOf(clientPKICheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether this component is configured to use local client VPN"+
" software to\n\t//access those remote components that require it"+
" (e.g., a VPN gateway).\n"+
"\tVPNClient: "+
String.valueOf(vPNClientCheckBox.isSelected()) +
" :end\n\n"+
"\t//Email server settings to determine if incoming e-mail"+
" attachments\n\t//are scanned for viruses\n\t//and/or are stripped.\n"+
"\tScanEmailAttachments: "+
String.valueOf(scanEmailCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether to strip email attachments,\n"+
"\tStripEmailAttachments: "+
String.valueOf(stripEmailCheckBox.isSelected()) +
" :end\n\n"+
"\t//Automatic lock/logout, determines if the workstation"+
" automatically requires\n\t//a password to restart activity after"+
" a timeout of the specified duration.\n\t//Boolean\n"+
"\tAutomaticLockLogout: "+
String.valueOf(autoLogoutCheckBox.isSelected()) +
" :end\n\n"+
```

```
"\t//Whether the non-MAC security attributes of this component are "+
"self\n\t//administered by the user who has access to the component."+
"\n\t//If \"true\", the other component configuration values are"+
" liable to \n\t//change based on user training, competence and"+
" trustworthiness.\n"+
"\tSelfAdminister: "+
String.valueOf(userAdminNoMACCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether the MAC security attributes of this component are"+
"self-administered.\n"+
"\tSelfAdministerMAC: "+
String.valueOf(userAdminMACCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether software can only be installed by administrator.\n"+
"\tAdministerSoftwareControl: "+
String.valueOf(adminOnlyInstallsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Block use of removable media and ports (e.g., USB)"+
"\n\t//(Affects ablility of malicious insider to extract sensitive"+
" data.  Also affects\n\t//chance of user introducing malicious"+
" software.)\n"+
"\tBlockRemovableMedia: "+
String.valueOf(blockMediaCheckBox.isSelected()) +
" :end\n\n"+
"\t//Block write access to local storage (Affects cost of theft of"+
" workstation, e.g.,\n\t//if the value is \"true\", then no assets"+
" are stored locally.\n"+
"\tBlockLocalStorage: "+
String.valueOf(blockWriteCheckBox.isSelected()) +
" :end\n\n"+
"\t//Browser settings. The most strict selected will be enacted\n"+
"\tBrowserSettingsLoose: "+
String.valueOf(browserLooseRadioButton.isSelected()) +
" :end\n"+
"\tBrowserSettingsNormal: "+
String.valueOf(browserNormalRadioButton.isSelected()) +
" :end\n"+
"\tBrowserSettingsStrict: "+
String.valueOf(browserStrictRadioButton.isSelected()) +
" :end\n\n"+
"\t//Email settings. The most strict will be enacted\n"+
"\tEmailSettingsLoose: "+
String.valueOf(emailLooseRadioButton.isSelected())+
" :end\n"+
"\tEmailSettingsNormal: "+
String.valueOf(emailNormalRadioButton.isSelected()) +
" :end\n"+
"\tEmailSettingsStrict: "+
String.valueOf(emailStrictRadioButton.isSelected()) +
" :end\n\n"+
"\t//Frequency at which patches are applied. The most strict will be" +
" enacted\n"+
"\tUpdatePatchesAsReleased: "+
String.valueOf(patchFreqAsReleasedRadioButton.isSelected())+
" :end\n"+
"\tUpdatePatchesRoutinely: "+
String.valueOf(patchFreqRoutinelyRadioButton.isSelected()) +
" :end\n"+
"\tUpdatePatchesAutomatically: "+
String.valueOf(patchFreqAutoRadioButton.isSelected()) +
" :end\n\n"+
"\t//Frequency of antivirus updates.\n" +
```

```
"\tUpdateAntivirusRegular: "+
String.valueOf(antiVirusRegRadioButton.isSelected()) +
" :end\n"+
"\tUpdateAntivirusAutomatic: "+
String.valueOf(antiVirusAutoRadioButton.isSelected()) +
" :end\n\n"+
"\t//**** The remaining fields are only meaningful if \"IsTemplate\" "+
"is \"false.\" ***\n\n"+
"\t//The zone that contains the component.\n"+
"\tZone: "+
zoneFinalTextField.getText() +
" :end\n\n"+
"\t//This is the name of the user that \"owns\" this computer.\n"+
"\tUser: "+
userFinalTextField.getText() +
" :end\n\n"+
"\t//Which workspace does this belong to.\n"+
"\tPosIndex: "+
posIndexTextField.getText() +
" :end\n\n"+
assetList_toString() +
localUserList_toString() +
remoteUserList_toString() +
authServerList_toString() +
"\t//Whether is component is to be connected to an uninteruptible"+
" power supply \n"+
"\tUninterruptiblePower: "+
String.valueOf(uPSCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether administrators routinely backup this component\n"+
"\tAdminBackup: "+
String.valueOf(regularBackupCheckBox.isSelected()) +
" :end\n\n"+
"\t//Whether backups are stored offsite\n"+
"\tOffsiteBackup: "+
String.valueOf(offSiteBackupsCheckBox.isSelected()) +
" :end\n\n"+
"\t//Determines the degree of configuration management applied to the"+
" component\n\t//and its software\n\t//The most restrictive \"true\""+
" setting applies:\n"+
"\tWeakCM: "+
String.valueOf(cMWeakRadioButton.isSelected()) +
" :end\n"+
"\tModerateCM: "+
String.valueOf(cMModerateRadioButton.isSelected()) +
" :end\n"+
"\tStrictCM: "+
String.valueOf(cMStrictRadioButton.isSelected()) +
" :end\n\n";
//add the selected network connections
if(network_added)
{
   Vector tempVec = new Vector();
   tempVec = networkConnectionFinalElementListVec;
   for(int i = 0; i < networkConnectionSourceFileListVec.size(); i++)
   {
      tempObj = new Object();
      ScenarioElementSet tempSet = new ScenarioElementSet();
      try
      {
```

```
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aText+"CyberCIEGE/SDT/Reusable Sets Library/Other/Component Network
Connection/"+networkConnectionSourceFileListVec.get(i).toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read-zone tostring",
                                   "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                      "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObj = tempSet.elementContainer.get(j);
                       NetworkConnection tempNetworkConnection = new NetworkConnection();
                       tempNetworkConnection = (NetworkConnection)tempObj;
                       for(int k = 0; k < tempVec.size(); k++)
                       {

if(tempNetworkConnection.getNameTextField().toString().equalsIgnoreCase(tempVec.get(k).toString()))
                          {
                             System.out.print(tempNetworkConnection.getNameTextField().toString()+"
"+tempVec.get(k).toString()+"\n");
                             outputString += tempNetworkConnection.toString(null);
                             tempVec.remove(k);
                          }
                       }
                    }
                 }
              }
              if(procedure_added)
              {
                 tempObj = new Object();
                 ScenarioElementSet tempSet = new ScenarioElementSet();
                 try
                 {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aText+"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural
Settings/"+mProceduralSettingFileName));
```

```
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                   JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file read-zone tostring",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   input.close();
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                   tempObj = tempSet.elementContainer.get(j);
                   ProceduralSettings tempProcSettings = new ProceduralSettings();
                   tempProcSettings = (ProceduralSettings)tempObj;

if(tempProcSettings.getNameTextField().toString().equalsIgnoreCase(procSettingsFinalTextField.getText()))
                   {
                      outputString += tempProcSettings.toString(null);
                   }
                }
             }
             outputString += "\n:end//Of The Component Section\n\n";
             return outputString;
          }

          //since the name field of the form is private this method provides a way
          //to fetch the name.
          public String getNameTextField()
          {
             return nameTextField.getText();
          }

          // Variables declaration - do not modify
          private javax.swing.JCheckBox adminOnlyInstallsCheckBox;
          private javax.swing.JRadioButton antiVirusAutoRadioButton;
          private javax.swing.JRadioButton antiVirusRegRadioButton;
          private javax.swing.ButtonGroup antiVirusbuttonGroup;
          private javax.swing.JCheckBox assetProtectionCheckBox;
          private javax.swing.JList assetsFinalList;
          private javax.swing.JScrollPane assetsFinalScrollPane;
```

```
private javax.swing.JLabel assetsLabel;
private javax.swing.JButton assetsMoveButton;
private javax.swing.JButton assetsRemoveButton;
private javax.swing.JList assetsSourceList;
private javax.swing.JScrollPane assetsSourceScrollPane;
private javax.swing.JList authFinalList;
private javax.swing.JScrollPane authFinalScrollPane;
private javax.swing.JButton authMoveButton;
private javax.swing.JButton authRemoveButton;
private javax.swing.JLabel authServListLabel;
private javax.swing.JList authSourceList;
private javax.swing.JScrollPane authSourceScrollPane;
private javax.swing.JCheckBox autoLogoutCheckBox;
private javax.swing.JPanel avPanel;
private javax.swing.JTextField availabilityTextField;
private javax.swing.JLabel availablityLabel;
private javax.swing.JCheckBox biometricsCheckBox;
private javax.swing.JCheckBox blockMediaCheckBox;
private javax.swing.JCheckBox blockWriteCheckBox;
private javax.swing.JPanel boolPanel;
private javax.swing.JRadioButton browserLooseRadioButton;
private javax.swing.JRadioButton browserNormalRadioButton;
private javax.swing.JPanel browserPanel;
private javax.swing.JRadioButton browserStrictRadioButton;
private javax.swing.ButtonGroup browserbuttonGroup;
private javax.swing.JRadioButton cMModerateRadioButton;
private javax.swing.JRadioButton cMStrictRadioButton;
private javax.swing.JRadioButton cMWeakRadioButton;
private javax.swing.JPanel checkboxPanel;
private javax.swing.JCheckBox clientPKICheckBox;
private javax.swing.JPanel cmPanel;
private javax.swing.JPanel comboBoxandTextPanel;
private javax.swing.ButtonGroup configMgmntbuttonGroup;
private javax.swing.JLabel costLabel;
private javax.swing.JTextField costTextField;
private javax.swing.JLabel descriptionLabel;
private javax.swing.JScrollPane descriptionScrollPane;
private javax.swing.JTextArea descriptionTextArea;
private javax.swing.JRadioButton emailLooseRadioButton;
private javax.swing.JRadioButton emailNormalRadioButton;
private javax.swing.JPanel emailPanel;
private javax.swing.JRadioButton emailStrictRadioButton;
private javax.swing.ButtonGroup emailbuttonGroup;
private javax.swing.JLabel hardwareNameLabel;
private javax.swing.JComboBox hardwareSourceComboBox;
private javax.swing.JPanel listsPanel;
private javax.swing.JList localUserFinalList;
private javax.swing.JScrollPane localUserFinalScrollPane;
private javax.swing.JLabel localUserLabel;
private javax.swing.JButton localUserMoveButton;
private javax.swing.JButton localUserRemoveButton;
private javax.swing.JList localUserSourceList;
private javax.swing.JScrollPane localUserSourceScrollPane;
private javax.swing.JLabel maintenanceLabel;
private javax.swing.JTextField maintenanceTextField;
private javax.swing.JPanel manyToOnePanel;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
private javax.swing.JList networkFinalList;
private javax.swing.JScrollPane networkFinalScrollPane;
private javax.swing.JLabel networkLabel;
private javax.swing.JButton networkMoveButton;
```

```
private javax.swing.JButton networkRemoveButton;
private javax.swing.JList networkSourceList;
private javax.swing.JScrollPane networkSourceScrollPane;
private javax.swing.JCheckBox offSiteBackupsCheckBox;
private javax.swing.JCheckBox oneTimePassCheckBox;
private javax.swing.JLabel opSysLabel;
private javax.swing.JComboBox opSysSourceComboBox;
private javax.swing.JCheckBox pKICertsCheckBox;
private javax.swing.JRadioButton patchFreqAsReleasedRadioButton;
private javax.swing.JRadioButton patchFreqAutoRadioButton;
private javax.swing.JRadioButton patchFreqRoutinelyRadioButton;
private javax.swing.ButtonGroup patchFreqbuttonGroup;
private javax.swing.JPanel patchesPanel;
private javax.swing.JPanel physcompPanel;
private javax.swing.JScrollPane physcompScrollPane;
private javax.swing.JComboBox posIndexFinalComboBox;
private javax.swing.JLabel posIndexLabel;
private javax.swing.JTextField posIndexTextField;
private javax.swing.JComboBox procSettingsComboBox;
private javax.swing.JTextField procSettingsFinalTextField;
private javax.swing.JLabel procSettingsLabel;
private javax.swing.JCheckBox regularBackupCheckBox;
private javax.swing.JCheckBox remoteAccessCheckBox;
private javax.swing.JList remoteUserFinalList;
private javax.swing.JScrollPane remoteUserFinalScrollPane;
private javax.swing.JLabel remoteUserLabel;
private javax.swing.JButton remoteUserMoveButton;
private javax.swing.JButton remoteUserRemoveButton;
private javax.swing.JList remoteUserSourceList;
private javax.swing.JScrollPane remoteUserSourceScrollPane;
private javax.swing.JLabel resaleLabel;
private javax.swing.JTextField resaleTextField;
private javax.swing.JCheckBox scanEmailCheckBox;
private javax.swing.JList softwareFinalList;
private javax.swing.JScrollPane softwareFinalScrollPane;
private javax.swing.JLabel softwareLabel;
private javax.swing.JButton softwareMoveButton;
private javax.swing.JButton softwareRemoveButton;
private javax.swing.JList softwareSourceList;
private javax.swing.JScrollPane softwareSourceScrollPane;
private javax.swing.JCheckBox stripEmailCheckBox;
private javax.swing.JCheckBox tokenPKICheckBox;
private javax.swing.JCheckBox uPSCheckBox;
private javax.swing.JCheckBox userAdminMACCheckBox;
private javax.swing.JCheckBox userAdminNoMACCheckBox;
private javax.swing.JTextField userFinalTextField;
private javax.swing.JLabel userLabel;
private javax.swing.JComboBox userSourceComboBox;
private javax.swing.JCheckBox vPNClientCheckBox;
private javax.swing.JTextField zoneFinalTextField;
private javax.swing.JLabel zoneLabel;
private javax.swing.JComboBox zonesourceComboBox;
// End of variables declaration
transient private java.io.ObjectInputStream input;
private boolean IsTemplate;
private boolean software_added;
private boolean assets_added;
private boolean trustedHost_added;
private boolean auth_server_added;
private boolean filter_added;
private boolean remoteUsers_added;
private boolean localUsers_added;
```

```
    private boolean network_added;
    private boolean procedure_added;
    private Vector  mSoftware_Vector;
    private Vector  mRemoteUserVector;
    private Vector  mLocalUserVector;
    private Vector  mAuthenticationVector;
    private Vector  mAssetVector;
    private Vector networkConnectionFinalFileListVec;
    private Vector networkConnectionFinalElementListVec;
    private Vector networkConnectionSourceFileListVec;
    private Vector networkConnectionSourceElementListVec;
    private String mProceduralSettingFileName;
    private Vector proceduralSettingsFileListVec;
    private Vector proceduralSettingsElementListVec;
    private Vector assetListVec;
    private Vector authServerListVec;
    private Vector userListVec;
    private Vector zoneListVec;
    private Scenario mScenario;
}
```

Q. THE SOURCE OF: PROCEDURAL SETTINGS

```
/*
 * ProceduralSettings.java
 *
 * Created on January 23, 2004, 11:19 AM
 */

package CCSDT;
import java.util.*;
import javax.swing.*;
import java.io.*;
/**
 *
 * @author  KJohns
 *This class is the graphical representation of a ProceduralSetting descriptor.
 *This is where the ProceduralSetting form is managed and the ProceduralSetting
 *toString() resides.
 */
public class ProceduralSettings ext ends ScenarioElement implements java.io.Serializable
{

    /** Creates new form ProceduralSettings */
    public ProceduralSettings()
    {
        initComponents();
        acl_added = false;
        aclTotalAccessListVec = new Vector();
        aclControlComboBox.addItem("Y");
        aclControlComboBox.addItem("N");
        aclControlComboBox.addItem("X");
        aclExecuteComboBox.addItem("Y");
        aclExecuteComboBox.addItem("N");
        aclExecuteComboBox.addItem("X");
        aclReadComboBox.addItem("Y");
        aclReadComboBox.addItem("N");
        aclReadComboBox.addItem("X");
        aclWriteComboBox.addItem("Y");
        aclWriteComboBox.addItem("N");
        aclWriteComboBox.addItem("X");
    }
```

```
private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;

    proceduralSettingsScrollPane = new javax.swing.JScrollPane();
    proceduralSettingsPanel = new javax.swing.JPanel();
    boxesPanel = new javax.swing.JPanel();
    nameLabel = new javax.swing.JLabel();
    nameTextField = new javax.swing.JTextField();
    boolPanel = new javax.swing.JPanel();
    holdsUserAssetCheckBox = new javax.swing.JCheckBox();
    noUseOfModemsCheckBox = new javax.swing.JCheckBox();
    noExternalSoftwareCheckBox = new javax.swing.JCheckBox();
    noEmailAttachmentExeCheckBox = new javax.swing.JCheckBox();
    lockOrLogoffCheckBox = new javax.swing.JCheckBox();
    writeDownPassCheckBox = new javax.swing.JCheckBox();
    protectWithACLCheckBox = new javax.swing.JCheckBox();
    noPhysicalModsCheckBox = new javax.swing.JCheckBox();
    noWebMailCheckBox = new javax.swing.JCheckBox();
    noMediaLeaveZoneCheckBox = new javax.swing.JCheckBox();
    updateAVCheckBox = new javax.swing.JCheckBox();
    applyPatchesCheckBox = new javax.swing.JCheckBox();
    leaveMachinesOnCheckBox = new javax.swing.JCheckBox();
    userBackupCheckBox = new javax.swing.JCheckBox();
    aclSecPanel = new javax.swing.JPanel();
    secIntSettingsPanel = new javax.swing.JPanel();
    mlMaxSecrecyLabel = new javax.swing.JLabel();
    mlMaxSecrecyComboBox = new javax.swing.JComboBox();
    mlMaxSecrecyFinalTextField = new javax.swing.JTextField();
    mlMinSecrecyLabel = new javax.swing.JLabel();
    mlMinSecrecyComboBox = new javax.swing.JComboBox();
    mlMinSecrecyFinalTextField = new javax.swing.JTextField();
    mlMaxIntegrityLabel = new javax.swing.JLabel();
    mlMaxIntegrityComboBox = new javax.swing.JComboBox();
    mlMaxIntegrityFinalTextField = new javax.swing.JTextField();
    mlMinIntegrityLabel = new javax.swing.JLabel();
    mlMinIntegrityComboBox = new javax.swing.JComboBox();
    mlMinIntegrityFinalTextField = new javax.swing.JTextField();
    aclPanel = new javax.swing.JPanel();
    aclUserLabel = new javax.swing.JLabel();
    aclGroupLabel = new javax.swing.JLabel();
    aclReadLabel = new javax.swing.JLabel();
    aclWriteLabel = new javax.swing.JLabel();
    aclControlLabel = new javax.swing.JLabel();
    aclExecuteLabel = new javax.swing.JLabel();
    aclUserComboBox = new javax.swing.JComboBox();
    aclUserFinalTextField = new javax.swing.JTextField();
    aclGroupComboBox = new javax.swing.JComboBox();
    aclGroupFinalTextField = new javax.swing.JTextField();
    aclReadComboBox = new javax.swing.JComboBox();
    aclWriteComboBox = new javax.swing.JComboBox();
    aclControlComboBox = new javax.swing.JComboBox();
    aclExecuteComboBox = new javax.swing.JComboBox();
    aclAddButton = new javax.swing.JButton();
    aclScrollPane = new javax.swing.JScrollPane();
    aclList = new javax.swing.JList();
    aclRemoveButton = new javax.swing.JButton();
    passwordPanel = new javax.swing.JPanel();
    passCharSetLabel = new javax.swing.JLabel();
    passLengthComboBox = new javax.swing.JComboBox();
    passLengthLabel = new javax.swing.JLabel();
    passCharSetComboBox = new javax.swing.JComboBox();
```

```
        passChangeFreqLabel = new javax.swing.JLabel();
        passChangeFreqComboBox = new javax.swing.JComboBox();

        setLayout(new java.awt.GridLayout(1, 0));

        setPreferredSize(new java.awt.Dimension(1026, 753));
        proceduralSettingsPanel.setLayout(new java.awt.GridBagLayout());

        proceduralSettingsPanel.setPreferredSize(new java.awt.Dimension(1025, 762));
        boxesPanel.setLayout(new java.awt.GridBagLayout());

        nameLabel.setLabelFor(nameTextField);
        nameLabel.setText("Procedural Settings Name:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        boxesPanel.add(nameLabel, gridBagConstraints);

        nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        boxesPanel.add(nameTextField, new java.awt.GridBagConstraints());

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        proceduralSettingsPanel.add(boxesPanel, gridBagConstraints);

        boolPanel.setLayout(new java.awt.GridBagLayout());

        boolPanel.setBorder(new javax.swing.border.TitledBorder(" Component Configuration Settings"));
        holdsUserAssetCheckBox.setText("Users private assets are stored on this machine");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        boolPanel.add(holdsUserAssetCheckBox, gridBagConstraints);

        noUseOfModemsCheckBox.setText("Users are instructed to not connect phone lines to modem ports");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 6;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        boolPanel.add(noUseOfModemsCheckBox, gridBagConstraints);

        noExternalSoftwareCheckBox.setText("Users are instructed to not introduce software from sources
external to the enterprise");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 5;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        boolPanel.add(noExternalSoftwareCheckBox, gridBagConstraints);

        noEmailAttachmentExeCheckBox.setText("Users are instructed to not open executable mail
attachments");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        boolPanel.add(noEmailAttachmentExeCheckBox, gridBagConstraints);

        lockOrLogoffCheckBox.setText("Workstation should be locked/logged off when user is not present");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
                boolPanel.add(lockOrLogoffCheckBox, gridBagConstraints);

                writeDownPassCheckBox.setText("Passwords may be written down and kept in accessible areas");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(writeDownPassCheckBox, gridBagConstraints);

                protectWithACLCheckBox.setText("Users should protect information on this mahine with ACLs");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 1;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(protectWithACLCheckBox, gridBagConstraints);

                noPhysicalModsCheckBox.setText("Users are instructed to not make any physical modifications to the
component");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 5;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(noPhysicalModsCheckBox, gridBagConstraints);

                noWebMailCheckBox.setText("Users are instructed to not use personal web mail from enterprise
systems");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 0;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(noWebMailCheckBox, gridBagConstraints);

                noMediaLeaveZoneCheckBox.setText("Users are instructed to not take media out of this zone");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 1;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(noMediaLeaveZoneCheckBox, gridBagConstraints);

                updateAVCheckBox.setText("Users are instructed to update antivirus software regularly");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(updateAVCheckBox, gridBagConstraints);

                applyPatchesCheckBox.setText("Users are instructed to apply security patches regularly");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 3;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(applyPatchesCheckBox, gridBagConstraints);

                leaveMachinesOnCheckBox.setText("Users leave workstations powered on to permit remote updates
and configuration");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 1;
                gridBagConstraints.gridy = 4;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                boolPanel.add(leaveMachinesOnCheckBox, gridBagConstraints);

                userBackupCheckBox.setText("Users are required to back up their own data on this component");
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel.add(userBackupCheckBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
proceduralSettingsPanel.add(boolPanel, gridBagConstraints);

aclSecPanel.setLayout(new java.awt.GridBagLayout());

aclSecPanel.setBorder(new javax.swing.border.TitledBorder("Allocation of Assets to Components"));
aclSecPanel.setPreferredSize(new java.awt.Dimension(747, 462));
secIntSettingsPanel.setLayout(new java.awt.GridBagLayout());

secIntSettingsPanel.setBorder(new javax.swing.border.TitledBorder("Allocation By Label"));
mlMaxSecrecyLabel.setText("Max Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secIntSettingsPanel.add(mlMaxSecrecyLabel, gridBagConstraints);

mlMaxSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMaxSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMaxSecrecyComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secIntSettingsPanel.add(mlMaxSecrecyComboBox, gridBagConstraints);

mlMaxSecrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
secIntSettingsPanel.add(mlMaxSecrecyFinalTextField, gridBagConstraints);

mlMinSecrecyLabel.setText("Min Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secIntSettingsPanel.add(mlMinSecrecyLabel, gridBagConstraints);

mlMinSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMinSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMinSecrecyComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secIntSettingsPanel.add(mlMinSecrecyComboBox, gridBagConstraints);

mlMinSecrecyFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
secIntSettingsPanel.add(mlMinSecrecyFinalTextField, gridBagConstraints);

mlMaxIntegrityLabel.setText("Max Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secIntSettingsPanel.add(mlMaxIntegrityLabel, gridBagConstraints);

mlMaxIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMaxIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMaxIntegrityComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secIntSettingsPanel.add(mlMaxIntegrityComboBox, gridBagConstraints);

mlMaxIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
secIntSettingsPanel.add(mlMaxIntegrityFinalTextField, gridBagConstraints);

mlMinIntegrityLabel.setText("Min Integrity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secIntSettingsPanel.add(mlMinIntegrityLabel, gridBagConstraints);

mlMinIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
mlMinIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      mlMinIntegrityComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secIntSettingsPanel.add(mlMinIntegrityComboBox, gridBagConstraints);

mlMinIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
secIntSettingsPanel.add(mlMinIntegrityFinalTextField, gridBagConstraints);

aclSecPanel.add(secIntSettingsPanel, new java.awt.GridBagConstraints());
```

```
aclPanel.setLayout(new java.awt.GridBagLayout());

aclPanel.setBorder(new javax.swing.border.TitledBorder("Allocation by Intended DAC"));
aclUserLabel.setLabelFor(aclUserComboBox);
aclUserLabel.setText("User");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
aclPanel.add(aclUserLabel, gridBagConstraints);

aclGroupLabel.setLabelFor(aclGroupComboBox);
aclGroupLabel.setText("Group");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
aclPanel.add(aclGroupLabel, gridBagConstraints);

aclReadLabel.setLabelFor(aclReadComboBox);
aclReadLabel.setText("Read");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
aclPanel.add(aclReadLabel, gridBagConstraints);

aclWriteLabel.setLabelFor(aclWriteComboBox);
aclWriteLabel.setText("Write");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
aclPanel.add(aclWriteLabel, gridBagConstraints);

aclControlLabel.setLabelFor(aclControlComboBox);
aclControlLabel.setText("Control");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 3;
aclPanel.add(aclControlLabel, gridBagConstraints);

aclExecuteLabel.setLabelFor(aclExecuteComboBox);
aclExecuteLabel.setText("Execute");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 3;
aclPanel.add(aclExecuteLabel, gridBagConstraints);

aclUserComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclUserComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclUserComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
aclPanel.add(aclUserComboBox, gridBagConstraints);

aclUserFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
```

```
aclPanel.add(aclUserFinalTextField, gridBagConstraints);

aclGroupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
aclGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclGroupComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
aclPanel.add(aclGroupComboBox, gridBagConstraints);

aclGroupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
aclPanel.add(aclGroupFinalTextField, gridBagConstraints);

aclReadComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
aclPanel.add(aclReadComboBox, gridBagConstraints);

aclWriteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
aclPanel.add(aclWriteComboBox, gridBagConstraints);

aclControlComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 4;
aclPanel.add(aclControlComboBox, gridBagConstraints);

aclExecuteComboBox.setPreferredSize(new java.awt.Dimension(100, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 5;
gridBagConstraints.gridy = 4;
aclPanel.add(aclExecuteComboBox, gridBagConstraints);

aclAddButton.setText("Add");
aclAddButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      aclAddButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclAddButton, gridBagConstraints);

aclScrollPane.setPreferredSize(new java.awt.Dimension(700, 150));
aclScrollPane.setViewportView(aclList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
```

```
gridBagConstraints.gridy = 7;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclScrollPane, gridBagConstraints);

aclRemoveButton.setText("Remove");
aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        aclRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
aclPanel.add(aclRemoveButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(20, 0, 20, 0);
aclSecPanel.add(aclPanel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
proceduralSettingsPanel.add(aclSecPanel, gridBagConstraints);

passwordPanel.setLayout(new java.awt.GridBagLayout());

passwordPanel.setBorder(new javax.swing.border.TitledBorder("Password Settings"));
passCharSetLabel.setLabelFor(passCharSetComboBox);
passCharSetLabel.setText("Password Character Set:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
passwordPanel.add(passCharSetLabel, gridBagConstraints);

passLengthComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
passwordPanel.add(passLengthComboBox, gridBagConstraints);

passLengthLabel.setLabelFor(passLengthComboBox);
passLengthLabel.setText("Password Length:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
passwordPanel.add(passLengthLabel, gridBagConstraints);

passCharSetComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
passwordPanel.add(passCharSetComboBox, gridBagConstraints);
```

```
        passChangeFreqLabel.setLabelFor(passChangeFreqComboBox);
        passChangeFreqLabel.setText("Password Change Frequency:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 7;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
        passwordPanel.add(passChangeFreqLabel, gridBagConstraints);

        passChangeFreqComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 7;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        passwordPanel.add(passChangeFreqComboBox, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        proceduralSettingsPanel.add(passwordPanel, gridBagConstraints);

        proceduralSettingsScrollPane.setViewportView(proceduralSettingsPanel);

        add(proceduralSettingsScrollPane);

    }
//the following listeners set a text field based on a combo box selection
    private void aclGroupComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclGroupComboBox.getSelectedItem() != null)
        {
            aclGroupFinalTextField.setText(
                    aclGroupComboBox.getSelectedItem().toString());
        }
    }

    private void aclUserComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(aclUserComboBox.getSelectedItem() != null)
        {
            aclUserFinalTextField.setText(
                    aclUserComboBox.getSelectedItem().toString());
        }
    }

    private void mlMinIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMinIntegrityComboBox.getSelectedItem() != null)
        {
            mlMinIntegrityFinalTextField.setText(
                    mlMinIntegrityComboBox.getSelectedItem().toString());
        }
    }

    private void mlMaxIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMaxIntegrityComboBox.getSelectedItem() != null)
        {
            mlMaxIntegrityFinalTextField.setText(
                    mlMaxIntegrityComboBox.getSelectedItem().toString());
        }
    }

    private void mlMinSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMinSecrecyComboBox.getSelectedItem() != null)
        {
            mlMinSecrecyFinalTextField.setText(
```

```
                    mlMinSecrecyComboBox.getSelectedItem().toString());
        }
    }

    private void mlMaxSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt) {
        if(mlMaxSecrecyComboBox.getSelectedItem() != null)
        {
            mlMaxSecrecyFinalTextField.setText(
                    mlMaxSecrecyComboBox.getSelectedItem().toString());
        }
    }
//the following listeners are for the buttons that either move or remove
    //entries from the acl and cost list, list boxes
    private void aclRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(aclList.getSelectedIndex() >= 0)
        {
            aclTotalAccessListVec.remove(aclList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)aclList.getModel();
            listModel.remove(aclList.getSelectedIndex());
            aclList = new JList(listModel);
            aclList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            aclScrollPane.setViewportView(aclList);
        }
    }

    private void aclAddButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        mACL = new AccessControlList();
        Object tempObject = new Object();
        AccessControlList tempACL = new AccessControlList();
        mACL.setUser(aclUserFinalTextField.getText());
        mACL.setGroup(aclGroupFinalTextField.getText());
        mACL.setRead(aclReadComboBox.getSelectedItem().toString());
        mACL.setWrite(aclWriteComboBox.getSelectedItem().toString());
        mACL.setControl(aclControlComboBox.getSelectedItem().toString());
        mACL.setExecute(aclExecuteComboBox.getSelectedItem().toString());
        aclTotalAccessListVec.addElement(mACL);
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < aclTotalAccessListVec.size(); i++)
        {
            tempObject = aclTotalAccessListVec.get(i);
            tempACL = (AccessControlList)tempObject;
            listModel.addElement(tempACL.getACL());
        }
        aclList = null;
        aclList = new JList(listModel);
        aclList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
        aclScrollPane.setViewportView(aclList);
        acl_added = true;
    }

    public void Load()
    {
    }

    public void Save()
    {
    }
```

```
//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
     // if the target of the vector is a list box just add the
     // vector directly
public void populateStaticSourceLists(java.util.Vector aPassComp,
      java.util.Vector aPassLen, java.util.Vector aPassChangeFreq)
{
   //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
   //to populate:
   //password complexity
   //password length
   //password change freq

   //password complexity
   for(int i = 0; i < aPassComp.size(); i++)
   {
      passCharSetComboBox.addItem(aPassComp.get(i));
   }
   //password length
   for(int i = 0; i < aPassLen.size(); i++)
   {
      passLengthComboBox.addItem(aPassLen.get(i));
   }
   //password change freq
   for(int i = 0; i < aPassLen.size(); i++)
   {
      passChangeFreqComboBox.addItem(aPassChangeFreq.get(i));
   }
}

//populateDynamicSourceLists() takes datapath and startupScenario as
//parameters in ScenarioDefinitionTool.java and is used to add dynamic
//content to combo boxes and list boxes.
//First the scenairo is used to get the added reusable sets
//Second any combo boxes to be used are cleared and reinitialized
//Third for each reusable set added a file input object is created
//Fourth for each member of the set the element name is added to the
//combo box or list.
public void populateDynamicSourceLists(String aPath, Scenario aScenario)
{
   //uses the Scenario passed to it to set the dynamic values of
   //dependent sets
   //User
   //DACGroup
   //Secrecy
   //Integrity
   mScenario = aScenario;
   Object tempObj = new Object();
   Vector tempVec = new Vector();
```

```
                ScenarioElementSet tempSet = new ScenarioElementSet();
                //User
                tempObj = aScenario.scenarioManager.get(11);
                tempVec = (Vector)tempObj;
                aclUserComboBox.removeAllItems();
                aclUserComboBox.addItem("*");
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   for(int j = 0; j < tempSet.elementContainer.size(); j++)
                   {
                      User tempUser = new User();
                      tempObj = tempSet.elementContainer.get(j);
                      tempUser = (User)tempObj;
                      aclUserComboBox.addItem(tempUser.getNameTextField());
                   }
                }
                //DACGroup
                tempObj = aScenario.scenarioManager.get(4);
                tempVec = (Vector)tempObj;
                aclGroupComboBox.removeAllItems();
                aclGroupComboBox.addItem("*");
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
```

```
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                   JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   input.close();
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file close",
                                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                   DACGroup tempDAC = new DACGroup();
                   tempObj = tempSet.elementContainer.get(j);
                   tempDAC = (DACGroup)tempObj;
                   aclGroupComboBox.addItem(tempDAC.getNameTextField());
                }
             }
             //Secrecy
             tempObj = aScenario.scenarioManager.get(9);
             tempVec = (Vector)tempObj;
             mlMaxSecrecyComboBox.removeAllItems();
             mlMinSecrecyComboBox.removeAllItems();
             for(int i = 0; i < tempVec.size(); i++)
             {
                tempObj = tempVec.get(i);
                try
                {
                   input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
```

```
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                    Secrecy tempSecrecy = new Secrecy();
                    tempObj = tempSet.elementContainer.get(j);
                    tempSecrecy = (Secrecy)tempObj;
                    mlMaxSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                    mlMinSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                }
            }
            //Integrity
            tempObj = aScenario.scenarioManager.get(6);
            tempVec = (Vector)tempObj;
            mlMaxIntegrityComboBox.removeAllItems();
            mlMinIntegrityComboBox.removeAllItems();
            for(int i = 0; i < tempVec.size(); i++)
            {
                tempObj = tempVec.get(i);
                try
                {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
```

```
            JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
        }
        try
        {
            input.close();
        }
        catch(IOException ioException)
        {
            JOptionPane.showMessageDialog(this, "Exception during file close",
                                "Exception", JOptionPane.ERROR_MESSAGE);
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
            Integrity tempIntegrity = new Integrity();
            tempObj = tempSet.elementContainer.get(j);
            tempIntegrity = (Integrity)tempObj;
            mlMaxIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
            mlMinIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
        }
    }
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
    DefaultListModel listModel = new DefaultListModel();
    Object tempObject = new Object();
    AccessControlList tempACL = new AccessControlList();
    for(int i = 0; i < aclTotalAccessListVec.size(); i++)
    {
        tempObject = aclTotalAccessListVec.get(i);
        tempACL = (AccessControlList)tempObject;
        listModel.addElement(tempACL.getACL());
    }
    aclList = null;
    aclList = new JList(listModel);
    aclList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
    aclScrollPane.setViewportView(aclList);
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
    aclAddButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            aclAddButtonActionPerformed(evt);
        }
    });

    aclRemoveButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            aclRemoveButtonActionPerformed(evt);
        }
    });
}
```

```
//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario,
      java.util.Vector aPassComp,java.util.Vector aPassLen,
      java.util.Vector aPassChangeFreq)
{
   populateDynamicSourceLists(aPath,aScenario);
   populateStaticSourceLists(aPassComp, aPassLen, aPassChangeFreq);

   mlMaxSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMaxSecrecyComboBoxActionPerformed(evt);
      }
   });
   mlMinSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMinSecrecyComboBoxActionPerformed(evt);
      }
   });
   mlMaxIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMaxIntegrityComboBoxActionPerformed(evt);
      }
   });
   mlMinIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         mlMinIntegrityComboBoxActionPerformed(evt);
      }
   });
   aclUserComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclUserComboBoxActionPerformed(evt);
      }
   });
   aclGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         aclGroupComboBoxActionPerformed(evt);
      }
   });
}

//the following name_toString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller name_toString() methods are called from the primary
//asset toString() method.

//creates the ACL list formatted output
private String aclList_toString()
{
   Object tempObject = new Object();
   AccessControlList tempACL = new AccessControlList();
   String aclString = new String();
   if(acl_added)
   {
      for(int i = 0; i < aclTotalAccessListVec.size(); i++)
      {
         tempObject = aclTotalAccessListVec.get(i);
         tempACL = (AccessControlList)tempObject;
         aclString +=
         "\t\tAccessList: " +
```

```
            tempACL.getUser() + "." + tempACL.getGroup()+
            " :end\tAccessMode: " +
            tempACL.getRead() + tempACL.getWrite() +
            tempACL.getControl() + tempACL.getExecute() +
            " :end\n";
        }
    }
    return aclString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an ProceduralSetting descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString =
    "\t\t//\"Holds user asset\" boolean determine whether user private"+
    " asset\n\t\t//pool should be stored on this machine\n"+
    "\t\tHoldsUserAsset: "+
    String.valueOf(holdsUserAssetCheckBox.isSelected())+
    " :end\n\n"+
    "\t\t//maximum secrecy of assets that should be created on this"+
    " component.\n"+
    "\t\tMaxSecrecyLabel: "+
    mlMaxSecrecyFinalTextField.getText()+
    " :end\n\n"+
    "\t\t//minimum secrecy of assets that should be created on this"+
    " component.\n"+
    "\t\tMimSecrecyLabel: "+
    mlMinSecrecyFinalTextField.getText()+
    " :end\n\n"+
    "\t\t//maximum integrity of assets that should be created on this"+
    " component.\n"+
    "\t\tMaxIntegrityyLabel: "+
    mlMaxIntegrityFinalTextField.getText()+
    " :end\n\n"+
    "\t\t//minimum integrity of assets that should be created on this"+
    " component.\n"+
    "\t\tMinIntegrityyLabel: "+
    mlMinIntegrityFinalTextField.getText()+
    " :end\n\n"+
    "\t\t//user and/or group and access mode of assets that should be"+
    " created on\n\t\t//this component.  Modes are read, write, control"+
    " and execute.\n\t\t//For example, an entry of \"*.finance YYXX\""+
    " means that\n\t\t//this component is intended to store assets"+
    " modified by\n\t\t//the finance group.\n\t\t//Available modes are"+
    " Y=Yes, N=No and X=Don't care\n"+
    aclList_toString()+
    "\n\t\t//User should protect information wiht ACLs.  If true, assets"+
    " on this\n\t\t//component should be protected via ACLs.\n"+
    "\t\tProtectWithACL: "+
    String.valueOf(protectWithACLCheckBox.isSelected())+
    " :end\n\n"+
    "\t\t//Boolean indicating whether passwords should be written down"+
    " in\n\t\t//areas where they might be accessible by others (e.g.,"+
    " unlocked\n\t\t//desk drawer)\n"+
    "\t\tWriteDownPasswords: "+
    String.valueOf(writeDownPassCheckBox.isSelected())+
    " :end\n\n"+
    "\t\t//Boolean indicating whether workstations should be locked or"+
```

```
" logged off\n\t\t//whenever the user is not physically present\n"+
"\t\tLockorLoggoff: "+
String.valueOf(lockOrLogoffCheckBox.isSelected())+
" :end\n\n"+
"\t\t//User instructed to use at least this length of password, value"+
" can\n\t\t//be \"long\", \"medium\", or \"short\"\n"+
"\t\tPasswordLength: ";
if(passLengthComboBox.getSelectedItem() != null)
{
   outputString += passLengthComboBox.getSelectedItem().toString();
}
outputString += " :end\n\n"+
"\t\t//The character sets that must be within the password, value"+
"\n\t\t//can be \"any\", \"moderate\", or \"complex\"\n"+
"\t\tPasswordCharacterSet: ";
if(passCharSetComboBox.getSelectedItem() != null)
{
   outputString += passCharSetComboBox.getSelectedItem().toString();
}
outputString +=" :end\n\n"+
"\t\t//The frequency of password changes, values can"+
"\n\t\t//be \"never\", \"twelve\", \"six\" or \"two\", (months)\n"+
"\t\tPasswordChangeFrequency: ";
if(passChangeFreqComboBox.getSelectedItem() != null)
{
   outputString += passChangeFreqComboBox.getSelectedItem().toString();
}
outputString += " :end\n\n"+
"\t\t//Users are instructed to not open executable email attachments\n"+
"\t\tNoEmailAttachmentsExecute: "+
String.valueOf(noEmailAttachmentExeCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Users are instructed to not introduce software from sources"+
" external to\n\t\t//the enterprise\n"+
"\t\tNoExternalSoftware: "+
String.valueOf(noExternalSoftwareCheckBox.isSelected())+
" :end\n\n"+
"\t\t//User are instructed to not connect phone lines to modem ports;\n"+
"\t\tNoUseOfModems: "+
String.valueOf(noUseOfModemsCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Users are instructed to not use personal web mail from "+
"enterprise\n\t\t//components\n"+
"\t\tNoWebMail: "+
String.valueOf(noWebMailCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Users are instructed to not take media out of this zone;\n"+
"\t\tNoMediaLeaveZone: "+
String.valueOf(noMediaLeaveZoneCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Users are instructed to update antivirus regularly\n"+
"\t\tUpdateAntiVirus: "+
String.valueOf(updateAVCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Users are instructed to apply security patches regularly\n"+
"\t\tApplyPatches: "+
String.valueOf(applyPatchesCheckBox.isSelected())+
" :end\n\n"+
"\t\t//Leave workstations powered on (to permit remote updates and"+
"\n\t\t//configuration)\n"+
"\t\tLeaveMachinesOn: "+
String.valueOf(leaveMachinesOnCheckBox.isSelected())+
```

```
    " :end\n\n"+
    "\t\t//Don't make any physical modifications to the component\n"+
    "\t\tNoPhysicalModifications: "+
    String.valueOf(noPhysicalModsCheckBox.isSelected())+
    " :end\n\n"+
    "\t\t//Whether user's are required to back up their own data on this"+
    " component\n"+
    "\t\tUserBackup: "+
    String.valueOf(userBackupCheckBox.isSelected())+
    " :end\n\n";
    return outputString;
}

//since the name field of the form is private this method provides a way
//to fetch the name.
public String getNameTextField()
{
    return nameTextField.getText();
}


// Variables declaration - do not modify
private javax.swing.JButton aclAddButton;
private javax.swing.JComboBox aclControlComboBox;
private javax.swing.JLabel aclControlLabel;
private javax.swing.JComboBox aclExecuteComboBox;
private javax.swing.JLabel aclExecuteLabel;
private javax.swing.JComboBox aclGroupComboBox;
private javax.swing.JTextField aclGroupFinalTextField;
private javax.swing.JLabel aclGroupLabel;
private javax.swing.JList aclList;
private javax.swing.JPanel aclPanel;
private javax.swing.JComboBox aclReadComboBox;
private javax.swing.JLabel aclReadLabel;
private javax.swing.JButton aclRemoveButton;
private javax.swing.JScrollPane aclScrollPane;
private javax.swing.JPanel aclSecPanel;
private javax.swing.JComboBox aclUserComboBox;
private javax.swing.JTextField aclUserFinalTextField;
private javax.swing.JLabel aclUserLabel;
private javax.swing.JComboBox aclWriteComboBox;
private javax.swing.JLabel aclWriteLabel;
private javax.swing.JCheckBox applyPatchesCheckBox;
private javax.swing.JPanel boolPanel;
private javax.swing.JPanel boxesPanel;
private javax.swing.JCheckBox holdsUserAssetCheckBox;
private javax.swing.JCheckBox leaveMachinesOnCheckBox;
private javax.swing.JCheckBox lockOrLogoffCheckBox;
private javax.swing.JComboBox mlMaxIntegrityComboBox;
private javax.swing.JTextField mlMaxIntegrityFinalTextField;
private javax.swing.JLabel mlMaxIntegrityLabel;
private javax.swing.JComboBox mlMaxSecrecyComboBox;
private javax.swing.JTextField mlMaxSecrecyFinalTextField;
private javax.swing.JLabel mlMaxSecrecyLabel;
private javax.swing.JComboBox mlMinIntegrityComboBox;
private javax.swing.JTextField mlMinIntegrityFinalTextField;
private javax.swing.JLabel mlMinIntegrityLabel;
private javax.swing.JComboBox mlMinSecrecyComboBox;
private javax.swing.JTextField mlMinSecrecyFinalTextField;
private javax.swing.JLabel mlMinSecrecyLabel;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
```

```
    private javax.swing.JCheckBox noEmailAttachmentExeCheckBox;
    private javax.swing.JCheckBox noExternalSoftwareCheckBox;
    private javax.swing.JCheckBox noMediaLeaveZoneCheckBox;
    private javax.swing.JCheckBox noPhysicalModsCheckBox;
    private javax.swing.JCheckBox noUseOfModemsCheckBox;
    private javax.swing.JCheckBox noWebMailCheckBox;
    private javax.swing.JComboBox passChangeFreqComboBox;
    private javax.swing.JLabel passChangeFreqLabel;
    private javax.swing.JComboBox passCharSetComboBox;
    private javax.swing.JLabel passCharSetLabel;
    private javax.swing.JComboBox passLengthComboBox;
    private javax.swing.JLabel passLengthLabel;
    private javax.swing.JPanel passwordPanel;
    private javax.swing.JPanel proceduralSettingsPanel;
    private javax.swing.JScrollPane proceduralSettingsScrollPane;
    private javax.swing.JCheckBox protectWithACLCheckBox;
    private javax.swing.JPanel secIntSettingsPanel;
    private javax.swing.JCheckBox updateAVCheckBox;
    private javax.swing.JCheckBox userBackupCheckBox;
    private javax.swing.JCheckBox writeDownPassCheckBox;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private Scenario mScenario;
    private boolean acl_added;
    private AccessControlList mACL;
    private Vector aclTotalAccessListVec;
}
```

R. THE SOURCE OF: SCENARIO

```
/*
 * Scenario.java
 *
 * Created on November 22, 2003, 1:23 PM
 */

package CCSDT;
import javax.swing.JOptionPane;
/**
 *
 * @author  kjohns
 *This class is the graphical representation of a Scenario.
 *This is where the Organization, Site, Briefing, DebriefWin and DebriefLose
 forms are managed and the  toString() for those descriptors resides.
 *
 *This is also where the scenarioManager data structure resides.  The scenario
 *manager is used to track the reusable sets that have been added to a sceanrio.
 *The tracking is accomplished by storing the file names of the sets in
 *vectors organized by descriptor type (all DACGroups together, all Assets
 *together etc.).  The reason file names are used to track the reusable sets is
 *because it makes the reusablity dynamic.  If an added reusable set is updated
 *the new information will appear in the next build of the scenario.  If copies
 *of reusable set object are stored in the scenairo manager then changes to the
 *file have no impact on the scenario.
 */

/*Regarding the scenarioManager structure
 *Index 0: Asset
 *Index 1: AssetGoal
 *Index 2: CatalogComponent
 *Index 3: Condition
 *Index 4: DACGroup
```

```
 *Index 5: Department
 *Index 6: Integrity
 *Index 7: Network
 *Index 8: PhysicalComponent
 *Index 9: Secrecy
 *Index 10: Trigger
 *Index 11: User
 *Index 12: Workspace
 *Index 13: Zone
 *Index 14: Other
 *         Index 0: Filter
 *         Index 1: ProceduralSettings
 *         Index 2: NetworkConnection
 */
public class Scenario extends ScenarioElement implements java.io.Serializable
{

   /** Creates new form Scenario */

   public Scenario(boolean aPrompt)
   {
      initComponents();
      if(aPrompt)
      {
         scenarioFileName = JOptionPane.showInputDialog(
                                 "Enter the scenario file name.")+
                                 ".CSM";
         scenarioDesigner = JOptionPane.showInputDialog(
                               "Enter the scenario designer name.");
      }

      //create the scenario manager and index it for 15.
      scenarioManager = new java.util.Vector(15);
      //add each of the descriptor vectors
      scenarioManager.add(0, new java.util.Vector());//Asset
      scenarioManager.add(1, new java.util.Vector());//AssetGoal
      scenarioManager.add(2, new java.util.Vector());//CatalogComponent
      scenarioManager.add(3, new java.util.Vector());//Condition
      scenarioManager.add(4, new java.util.Vector());//DACGroup
      scenarioManager.add(5, new java.util.Vector());//Department
      scenarioManager.add(6, new java.util.Vector());//Integrity
      scenarioManager.add(7, new java.util.Vector());//Network
      scenarioManager.add(8, new java.util.Vector());//PhysicalComponent
      scenarioManager.add(9, new java.util.Vector());//Secrecy
      scenarioManager.add(10, new java.util.Vector());//Trigger
      scenarioManager.add(11, new java.util.Vector());//User
      scenarioManager.add(12, new java.util.Vector());//Workspace
      scenarioManager.add(13, new java.util.Vector());//Zone
      //the 14th index is where the secondary-descriptors are held
      scenarioManager.add(14, new java.util.Vector());//Other
      Object tempObj = scenarioManager.get(14);
      java.util.Vector tempSet = (java.util.Vector)tempObj;
      tempSet.add(0, new java.util.Vector());//Filter
      tempSet.add(1, new java.util.Vector());//ProceduralSettings
      tempSet.add(2, new java.util.Vector());//NetworkConnection

      startmonthComboBox.addItem("January");
      startmonthComboBox.addItem("February");
      startmonthComboBox.addItem("March");
      startmonthComboBox.addItem("April");
      startmonthComboBox.addItem("May");
      startmonthComboBox.addItem("June");
```

```
    startmonthComboBox.addItem("July");
    startmonthComboBox.addItem("August");
    startmonthComboBox.addItem("September");
    startmonthComboBox.addItem("October");
    startmonthComboBox.addItem("November");
    startmonthComboBox.addItem("December");
    startdayComboBox.setVisible(false);
}

private void initComponents() {
    java.awt.GridBagConstraints gridBagConstraints;

    scenarioScrollPane = new javax.swing.JScrollPane();
    scenarioPanel = new javax.swing.JPanel();
    orgnameLabel = new javax.swing.JLabel();
    orgnameTextField = new javax.swing.JTextField();
    scenariotitleLabel = new javax.swing.JLabel();
    scenariotitleTextField = new javax.swing.JTextField();
    orgtypeLabel = new javax.swing.JLabel();
    orgtypeComboBox = new javax.swing.JComboBox();
    startdateLabel = new javax.swing.JLabel();
    startmonthComboBox = new javax.swing.JComboBox();
    startdayComboBox = new javax.swing.JComboBox();
    startmoneyLabel = new javax.swing.JLabel();
    budgetLabel = new javax.swing.JLabel();
    profitsharingLabel = new javax.swing.JLabel();
    sitenameLabel = new javax.swing.JLabel();
    sitenameTextField = new javax.swing.JTextField();
    floorplanLabel = new javax.swing.JLabel();
    floorplanComboBox = new javax.swing.JComboBox();
    isstaticCheckBox = new javax.swing.JCheckBox();
    cityLabel = new javax.swing.JLabel();
    cityTextField = new javax.swing.JTextField();
    stateLabel = new javax.swing.JLabel();
    stateTextField = new javax.swing.JTextField();
    sitedescriptionLabel = new javax.swing.JLabel();
    briefingLabel = new javax.swing.JLabel();
    debriefwinLabel = new javax.swing.JLabel();
    debrieflose Label = new javax.swing.JLabel();
    budgetTextField = new javax.swing.JTextField();
    startmoneyTextField = new javax.swing.JTextField();
    profitsharingTextField = new javax.swing.JTextField();
    sitedescriptionScrollPane = new javax.swing.JScrollPane();
    sitedescriptionTextArea = new javax.swing.JTextArea();
    briefingScrollPane = new javax.swing.JScrollPane();
    briefingTextArea = new javax.swing.JTextArea();
    debriefwinScrollPane = new javax.swing.JScrollPane();
    debriefwinTextArea = new javax.swing.JTextArea();
    debrieflose ScrollPane = new javax.swing.JScrollPane();
    debrieflose TextArea = new javax.swing.JTextArea();

    setLayout(new java.awt.GridLayout(1, 0));

    scenarioPanel.setLayout(new java.awt.GridBagLayout());

    orgnameLabel.setLabelFor(orgnameTextField);
    orgnameLabel.setText("Organization Name:");
    gridBagConstraints = new java.awt.GridBagConstraints();
    gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
    scenarioPanel.add(orgnameLabel, gridBagConstraints);

    orgnameTextField.setPreferredSize(new java.awt.Dimension(150, 19));
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(orgnameTextField, gridBagConstraints);

scenariotitleLabel.setLabelFor(scenariotitleTextField);
scenariotitleLabel.setText("Scenario Title:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(scenariotitleLabel, gridBagConstraints);

scenariotitleTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(scenariotitleTextField, gridBagConstraints);

orgtypeLabel.setLabelFor(orgtypeComboBox);
orgtypeLabel.setText("Organization Type:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(orgtypeLabel, gridBagConstraints);

orgtypeComboBox.setPreferredSize(new java.awt.Dimension(150, 24));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(orgtypeComboBox, gridBagConstraints);

startdateLabel.setLabelFor(startmonthComboBox);
startdateLabel.setText("Start Date:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(startdateLabel, gridBagConstraints);

startmonthComboBox.setPreferredSize(new java.awt.Dimension(90, 24));
startmonthComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        startmonthComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(startmonthComboBox, gridBagConstraints);

startdayComboBox.setPreferredSize(new java.awt.Dimension(50, 25));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(startdayComboBox, gridBagConstraints);
```

```
startmoneyLabel.setLabelFor(startmoneyTextField);
startmoneyLabel.setText("Start Up Money:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
gridBagConstraints.insets = new java.awt.Insets(0, 6, 0, 0);
scenarioPanel.add(startmoneyLabel, gridBagConstraints);

budgetLabel.setLabelFor(budgetTextField);
budgetLabel.setText("Budget:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(budgetLabel, gridBagConstraints);

profitsharingLabel.setLabelFor(profitsharingTextField);
profitsharingLabel.setText("Profit Sharing:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(profitsharingLabel, gridBagConstraints);

sitenameLabel.setLabelFor(sitenameTextField);
sitenameLabel.setText("Site Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(sitenameLabel, gridBagConstraints);

sitenameTextField.setPreferredSize(new java.awt.Dimension(150, 19));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(sitenameTextField, gridBagConstraints);

floorplanLabel.setLabelFor(floorplanComboBox);
floorplanLabel.setText("Floor Plan:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(floorplanLabel, gridBagConstraints);

floorplanComboBox.setPreferredSize(new java.awt.Dimension(150, 25));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
scenarioPanel.add(floorplanComboBox, gridBagConstraints);

isstaticCheckBox.setSelected(true);
isstaticCheckBox.setText("Player cannot change the attributes of components within the site:");
isstaticCheckBox.setHorizontalTextPosition(javax.swing.SwingConstants.LEADING);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(isstaticCheckBox, gridBagConstraints);

cityLabel.setLabelFor(cityTextField);
cityLabel.setText("City:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(cityLabel, gridBagConstraints);

cityTextField.setPreferredSize(new java.awt.Dimension(150, 19));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(cityTextField, gridBagConstraints);

stateLabel.setLabelFor(stateTextField);
stateLabel.setText("State:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(stateLabel, gridBagConstraints);

stateTextField.setPreferredSize(new java.awt.Dimension(75, 19));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(stateTextField, gridBagConstraints);

sitedescriptionLabel.setText("Site Description:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(sitedescriptionLabel, gridBagConstraints);

briefingLabel.setText("Start Up Briefing:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(briefingLabel, gridBagConstraints);

debriefwinLabel.setText("Win Message:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(debriefwinLabel, gridBagConstraints);

debrieflose Label.setText("Lose Message:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
scenarioPanel.add(debrieflose Label, gridBagConstraints);

budgetTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
scenarioPanel.add(budgetTextField, gridBagConstraints);

startmoneyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
```

```
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        scenarioPanel.add(startmoneyTextField, gridBagConstraints);

        profitsharingTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        scenarioPanel.add(profitsharingTextField, gridBagConstraints);

        sitedescriptionTextArea.setPreferredSize(new java.awt.Dimension(450, 100));
        sitedescriptionScrollPane.setViewportView(sitedescriptionTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 8;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        scenarioPanel.add(sitedescriptionScrollPane, gridBagConstraints);

        briefingTextArea.setPreferredSize(new java.awt.Dimension(450, 100));
        briefingScrollPane.setViewportView(briefingTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        scenarioPanel.add(briefingScrollPane, gridBagConstraints);

        debriefwinTextArea.setPreferredSize(new java.awt.Dimension(450, 50));
        debriefwinScrollPane.setViewportView(debriefwinTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 12;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        scenarioPanel.add(debriefwinScrollPane, gridBagConstraints);

        debriefloseTextArea.setPreferredSize(new java.awt.Dimension(450, 50));
        debriefloseScrollPane.setViewportView(debriefloseTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 14;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        scenarioPanel.add(debriefloseScrollPane, gridBagConstraints);

        scenarioScrollPane.setViewportView(scenarioPanel);

        add(scenarioScrollPane);

    }
//this combo box is used to set the start month and day.  Based on the
    //month selected the day combo box will be populated with the correct
    //number of days for that month
    private void startmonthComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
            // Add your handling code here:
        switch(startmonthComboBox.getSelectedIndex())
        {
           case 0://january
```

```
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
}
case 1://february
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   break;
}
case 2://march
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
```

```
}
case 3://april
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox. addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   break;
}
case 4://may
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startday ComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
}
case 5://june
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
```

```
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   break;
}
case 6://july
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
}
case 7://august
{
   startday ComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startday ComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
}
case 8://september
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
```

```
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");;
   break;
}
case 9://october
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   startdayComboBox.addItem("31");
   break;
}
case 10://november
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
   startdayComboBox.addItem("19");startdayComboBox.addItem("20");
   startdayComboBox.addItem("21");startdayComboBox.addItem("22");
   startdayComboBox.addItem("23");startdayComboBox.addItem("24");
   startdayComboBox.addItem("25");startdayComboBox.addItem("26");
   startdayComboBox.addItem("27");startdayComboBox.addItem("28");
   startdayComboBox.addItem("29");startdayComboBox.addItem("30");
   break;
}
case 11://december
{
   startdayComboBox.setVisible(true);
   startdayComboBox.removeAllItems();
   startdayComboBox.addItem("1");startdayComboBox.addItem("2");
   startdayComboBox.addItem("3");startdayComboBox.addItem("4");
   startdayComboBox.addItem("5");startdayComboBox.addItem("6");
   startdayComboBox.addItem("7");startdayComboBox.addItem("8");
   startdayComboBox.addItem("9");startdayComboBox.addItem("10");
   startdayComboBox.addItem("11");startdayComboBox.addItem("12");
   startdayComboBox.addItem("13");startdayComboBox.addItem("14");
   startdayComboBox.addItem("15");startdayComboBox.addItem("16");
   startdayComboBox.addItem("17");startdayComboBox.addItem("18");
```

```
            startdayComboBox.addItem("19");startdayComboBox.addItem("20");
            startdayComboBox.addItem("21");startdayComboBox.addItem("22");
            startdayComboBox.addItem("23");startdayComboBox.addItem("24");
            startdayComboBox.addItem("25");startdayComboBox.addItem("26");
            startdayComboBox.addItem("27");startdayComboBox.addItem("28");
            startdayComboBox.addItem("29");startdayComboBox.addItem("30");
            startdayComboBox.addItem("31");
            break;
         }
         default:
         {
            startdayComboBox.setVisible(false);
            startdayComboBox.removeAllItems();
            break;
         }
      }
   }

   public void setScenarioFileName(java.lang.String aSceName)
   {
      scenarioFileName = aSceName;
   }

   public String getScenarioFileName()
   {
      return scenarioFileName;
   }

   public void setDesignerName(java.lang.String aDesName)
   {
      scenarioDesigner = aDesName;
   }

   public String getDesignerName()
   {
      return scenarioDesigner;
   }

   //Interface combo box and list box content comes from one of two sources
   //the content is either from ini files or from reusable sets that have
   //already been added to the scenario in development.
   //If the content come from ini files it is designated static.  This is
   //because the content of ini files does not change.
   //If the content comes from reusable sets that have been added to the
   //scenario in development it is designated dynamic.  This is because
   //the content may be empty, of any lenght and change frequently

   //populateStaticSourceLists() takes a list of vectors as
   //parameters in ScenarioDefinitionTool.java and is used to add static
   //content to combo boxes and list boxes.
   //First any combo boxes are cleared and reinitialized
   //Second if the target of the vector is a combo box - for each
   //element in the vector add it to the combo box
        // if the target of the vector is a list box just add the
        // vector directly
   public void populateStaticSourceLists(java.util.Vector aFloorplan,
                                         java.util.Vector aOrganization)
   {
      //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
      //to populate:
      //orgtype
      //floorplan
```

```
   //floorplan
   for(int i = 0; i < aFloorplan.size(); i++)
   {
      floorplanComboBox.addItem(aFloorplan.get(i));
   }

   //organization
   for(int i = 0; i < aOrganization.size(); i++)
   {
      orgtypeComboBox.addItem(aOrganization.get(i));
   }
}

public void Save()
{

}

public void Load()
{

}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//since scenario is the keeper of the non-reusable descriptors its approach
//to toString() had to be slightly different.  The descriptors that scenario
//manages appear at the beinning and near the end of an SDF.  To deal with
//this there are 2 toString() methods.  One for the beginning of the
//SDF and one for the end.

//creates the formatted output for the organization and site descriptors
public String toString(String aText)
{
   String outputString = new String();
   outputString =
   "Organization:\n"+
   "\t//A text string to use as the name of the organization in this"+
   " scenario\n"+
   "\tName: " + orgnameTextField.getText() + " :end\n\n" +
   "\t//A text string name for the scenario\n"+
   "\tTitle: " + scenariotitleTextField.getText() + " :end\n\n" +
   "\t//Example type enumerations include Corporate, Government, Small"+
   "\n\t//Business, University, Home, etc.\n"+
   "\tType: " + orgtypeComboBox.getSelectedItem().toString() +
   " :end\n\n" +
   "\t//the starting calendar date of the scenario in the form"+
   "\n\t//Month (integer) Date (integer)\n"+
   "\tStartDate: " +
   String.valueOf(startmonthComboBox.getSelectedIndex()+1) + " " +
   String.valueOf(startdayComboBox.getSelectedIndex()+1)+ " :end\n\n" +
   "\t//An integer value specifying the amount of starting capital the"+
   "player\n\t//starts this scenario with\n"+
   "\tStartMoney: " + startmoneyTextField.getText() + " :end\n\n" +
   "\t//An integer value specifying the monthly budget of the IT dept\n"+
   "\tBudget: " + budgetTextField.getText() + " :end\n\n" +
   "\t//A scale factor from 0 (meaning none) to 100 (meaning max) "+
   "specifying how\n\t//much of the overall company's profits flow back"+
   " into the IT budget.\n"+
   "\tProfitSharing: " + profitsharingTextField.getText() + " :end\n\n" +
```

```
    ":end //of Organization\n\n" +
    "Site:\n" +
    "\t//A text string to use as a site tag in this scenario file\n"+
    "\tName: " + sitenameTextField.getText() + " :end\n\n" +
    "\t//Enumeration specifying which office floor plan to use for this"+
    " site\n"+
    "\tFloorPlan: " + floorplanComboBox.getSelectedItem().toString() +
    " :end\n\n" +
    "\t//If IsStatic is set to true it means that the player cannot"+
    " change the attributes of\n\t//components within the site\n"+
    "\tIsStatic: " + String.valueOf(isstaticCheckBox.isSelected()) +
    " :end\n\n" +
    "\t//Used for descriptive text in the game, as well as to support"+
    " multi-site scenarios\n"+
    "\tDescription: " + sitedescriptionTextArea.getText() + " :end\n\n" +
    "\t//Name of the city and state\n"+
    "\tCity: " + cityTextField.getText() + " :end\n\n" +
    "\tState: " + stateTextField.getText() + " :end\n\n" +
    ":end  //of Site\n\n";
    return outputString;
}

//creates the formatted output for the briefing, debrief win and debrief
//lose descriptors
public String toStringPost(String aText)
{
    String outputString = new String();
    outputString =
    "Briefing:\n"+
    "\t" + briefingTextArea.getText() + "\n" +
    ":end\n\n" +
    "DebriefWin:\n" +
    "\t" + debriefwinTextArea.getText() + "\n" +
    ":end\n\n"+
    "DebriefLose:\n" +
    "\t" + debrieflosetextArea.getText() + "\n" +
    ":end\n\n";
    return outputString;
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners()
{

    startmonthComboBox.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            startmonthComboBoxActionPerformed(evt);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JLabel briefingLabel;
private javax.swing.JScrollPane briefingScrollPane;
private javax.swing.JTextArea briefingTextArea;
private javax.swing.JLabel budgetLabel;
private javax.swing.JTextField budgetTextField;
private javax.swing.JLabel cityLabel;
```

```
    private javax.swing.JTextField cityTextField;
    private javax.swing.JLabel debriefloseLabel;
    private javax.swing.JScrollPane debriefloseScrollPane;
    private javax.swing.JTextArea debriefloseTextArea;
    private javax.swing.JLabel debriefwinLabel;
    private javax.swing.JScrollPane debriefwinScrollPane;
    private javax.swing.JTextArea debriefwinTextArea;
    private javax.swing.JComboBox floorplanComboBox;
    private javax.swing.JLabel floorplanLabel;
    private javax.swing.JCheckBox isstaticCheckBox;
    private javax.swing.JLabel orgnameLabel;
    private javax.swing.JTextField orgnameTextField;
    private javax.swing.JComboBox orgtypeComboBox;
    private javax.swing.JLabel orgtypeLabel;
    private javax.swing.JLabel profitsharingLabel;
    private javax.swing.JTextField profitsharingTextField;
    private javax.swing.JPanel scenarioPanel;
    private javax.swing.JScrollPane scenarioScrollPane;
    private javax.swing.JLabel scenariotitleLabel;
    private javax.swing.JTextField scenariotitleTextField;
    private javax.swing.JLabel sitedescriptionLabel;
    private javax.swing.JScrollPane sitedescriptionScrollPane;
    private javax.swing.JTextArea sitedescriptionTextArea;
    private javax.swing.JLabel sitenameLabel;
    private javax.swing.JTextField sitenameTextField;
    private javax.swing.JLabel startdateLabel;
    private javax.swing.JComboBox startdayComboBox;
    private javax.swing.JLabel startmoneyLabel;
    private javax.swing.JTextField startmoneyTextField;
    private javax.swing.JComboBox startmonthComboBox;
    private javax.swing.JLabel stateLabel;
    private javax.swing.JTextField stateTextField;
    // End of variables declaration
    private java.lang.String scenarioFileName;
    private java.lang.String scenarioDesigner;
    public java.util.Vector scenarioManager;//a vector to store
                                    //sets of sets of elements
}
```

S. THE SOURCE OF: SCENARIO DEFINITION TOOL

```
/*
 * ScenarioDefinitionTool.java
 *
 * Created on November 19, 2003, 1:47 PM
 */

package CCSDT;

import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.awt.dnd.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.tree.*;
import javax.swing.filechooser.*;
import javax.swing.JOptionPane;

/**
 *
```

```
 * @author  kjohns
 *
 *This class provides the main logic for the scenario definiion tool and
 *provides the foudation for the application.
 *Here is a list of the descriptors and their SDT file extensions
 *File extensions are named in the following way:
 *For a descriptor it is some letter identifying the descriptor followed by MS
 *A scenario is named with .CSM = CyberCIEGE Scenario Master
 *                Asset - .AMS
 *            AssetGoal - .GMS
 *       CatalogComponent - .CMS
 *            Condition - .BMS
 *             DACGroup - .DMS
 *            Department - .EMS
 *              Filter - .FMS
 *            Integrity - .IMS
 *              Network - .NMS
 *      NetworkConnection - .KMS
 *      PhysicalComponent - .PMS
 *     ProceduralSettings - .LMS
 *              Scenario - .CSM
 *               Secrecy - .SMS
 *              Trigger - .TMS
 *                User - .UMS
 *            Workspace - .WMS
 *                Zone - .ZMS
 *
 */
public class ScenarioDefinitionTool extends javax.swing.JFrame
{

    /** Creates new form ScenarioDefinitionTool */
    public ScenarioDefinitionTool()
    {
        //read in the static information used in the game from the
        //local ini files
        readINIFiles();
        //create a new scenairo and with it the scenario manager
        startupScenario = new Scenario(true);
        //creates the tab manager - a vector used to store the objects
        //presented in the tabbed work area
        tabManager = new java.util.Vector();
        //the scenairo is ALWAYS the first tab and index 0 of the
        //tab manager
        tabManager.insertElementAt(startupScenario, 0);
        //sets up the GUI
        initComponents();
        //Adds the start up scenario to the tabbed work area
        workareaTabbedPane.addTab("Scenario:" +
                    startupScenario.getScenarioFileName(), startupScenario);
        //initialize the feedback area to be blank
        feedbackTextArea.setText(null);
        //set up the variables used to delete set elements
        deleteTabTarget = -1;
        deleteElementTarget = -1;
        //initialize the string used to store the tree path for right clicks
        rightClickTreePath = new String();
        //initialize the static contents (using some of the data from
        //readINIfiles()) of the scenario
        startupScenario.populateStaticSourceLists(fileContents_floorplan,
                                        fileContents_organization);
    }
```

```
            private void initComponents() {
                libraryTreePopupMenu = new javax.swing.JPopupMenu();
                ltOpenMenuItem = new javax.swing.JMenuItem();
                ltaddMenuItem = new javax.swing.JMenuItem();
                ltDeleteMenuItem = new javax.swing.JMenuItem();
                scenarioTreePopupMenu = new javax.swing.JPopupMenu();
                stOpenMenuItem = new javax.swing.JMenuItem();
                stRemoveMenuItem = new javax.swing.JMenuItem();
                workAreaTabPopupMenu = new javax.swing.JPopupMenu();
                waAddMenuItem = new javax.swing.JMenuItem();
                waRemoveMenuItem = new javax.swing.JMenuItem();
                sdtToolBar = new javax.swing.JToolBar();
                jPanel1 = new javax.swing.JPanel();
                setelementmanagementLabel = new javax.swing.JLabel();
                setelementmanagementComboBox = new javax.swing.JComboBox();
                setelementdeleteButton = new javax.swing.JButton();
                feedbackareaScrollPane = new javax.swing.JScrollPane();
                feedbackTextArea = new javax.swing.JTextArea();
                treePanel = new javax.swing.JPanel();
                libraryScrollPane = new javax.swing.JScrollPane();
                libraryTree =  new javax.swing.JTree(librarytreePopulate());

libraryTree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
                DragSource libraryTreeDragSource = DragSource.getDefaultDragSource();
                libraryTreeDragSource.createDefaultDragGestureRecognizer(libraryTree,
DnDConstants.ACTION_COPY,
                    new DragGestureListener()
                    {
                       public void dragGestureRecognized(DragGestureEvent event)
                       {
                          JTree tempTree = (JTree)event.getComponent();
                          String tempString = tempTree.getSelectionPath().toString();
                          ScenarioElementSet draggedValue = getDragSourceFileFromTree(tempString);
                          Transferable transferable = new ScenarioElementSetTransferable(draggedValue);
                          event.startDrag(null,transferable,new ScenarioElementSetDragSourceListener());
                       }
                    });
                    scenarioScrollPane = new javax.swing.JScrollPane();
                    scenarioTree = new javax.swing.JTree(scenariotreePopulate());

scenarioTree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
                    scenarioTreeTarget = new DropTarget(scenarioTree,new
ScenarioElementSetDropTargetListener(startupScenario, scenarioTree));
                    workareaTabbedPane = new javax.swing.JTabbedPane();
                    sdtMenuBar = new javax.swing.JMenuBar();
                    fileMenu = new javax.swing.JMenu();
                    newsubMenu = new javax.swing.JMenu();
                    assetMenuItem = new javax.swing.JMenuItem();
                    assetGoalMenuItem = new javax.swing.JMenuItem();
                    catalogcompMenuItem = new javax.swing.JMenuItem();
                    conditionMenuItem = new javax.swing.JMenuItem();
                    dacGroupMenuItem = new javax.swing.JMenuItem();
                    departmentMenuItem = new javax.swing.JMenuItem();
                    integrityMenuItem = new javax.swing.JMenuItem();
                    networkMenuItem = new javax.swing.JMenuItem();
                    otherMenu = new javax.swing.JMenu();
                    filterMenuItem = new javax.swing.JMenuItem();
                    networkConnectionMenuItem = new javax.swing.JMenuItem();
                    proceduralSettingsMenuItem = new javax.swing.JMenuItem();
                    physcompMenuItem = new javax.swing.JMenuItem();
                    secrecyMenuItem = new javax.swing.JMenuItem();
```

```
triggerMenuItem = new javax.swing.JMenuItem();
userMenuItem = new javax.swing.JMenuItem();
workspaceMenuItem = new javax.swing.JMenuItem();
zoneMenuItem = new javax.swing.JMenuItem();
openMenuItem = new javax.swing.JMenuItem();
jSeparator1 = new javax.swing.JSeparator();
saveMenuItem = new javax.swing.JMenuItem();
savescenarioMenuItem = new javax.swing.JMenuItem();
saveallMenuItem = new javax.swing.JMenuItem();
saveasMenuItem = new javax.swing.JMenuItem();
savescenarioasMenuItem = new javax.swing.JMenuItem();
jSeparator2 = new javax.swing.JSeparator();
exitMenuItem = new javax.swing.JMenuItem();
toolsMenu = new javax.swing.JMenu();
buildMenuItem = new javax.swing.JMenuItem();

ltOpenMenuItem.setText("Open");
ltOpenMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      ltOpenMenuItemActionPerformed(evt);
   }
});

libraryTreePopupMenu.add(ltOpenMenuItem);

ltaddMenuItem.setText("Add to Scenario");
ltaddMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      ltaddMenuItemActionPerformed(evt);
   }
});

libraryTreePopupMenu.add(ltaddMenuItem);

ltDeleteM enuItem.setText("Delete");
ltDeleteMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      ltDeleteMenuItemActionPerformed(evt);
   }
});

libraryTreePopupMenu.add(ltDeleteMenuItem);

stOpenMenuItem.setText("Open");
stOpenMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      stOpenMenuItemActionPerformed(evt);
   }
});

scenarioTreePopupMenu.add(stOpenMenuItem);

stRemoveMenuItem.setText("Remove From Scenario");
stRemoveMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      stRemoveMenuItemActionPerformed(evt);
   }
});

scenarioTreePopupMenu.add(stRemoveMenuItem);

waAddMenuItem.setText("Add Tab to Scenario");
```

```
waAddMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      waAddMenuItemActionPerformed(evt);
   }
});

workAreaTabPopupMenu.add(waAddMenuItem);

waRemoveMenuItem.setText("Remove Tab");
waRemoveMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      waRemoveMenuItemActionPerformed(evt);
   }
});

workAreaTabPopupMenu.add(waRemoveMenuItem);

setTitle("CyberCIEGE Scenario Definition Tool");
setName("sdtframe");
addWindowListener(new java.awt.event.WindowAdapter() {
   public void windowClosing(java.awt.event.WindowEvent evt) {
      exitForm(evt);
   }
});

jPanel1.setLayout(new java.awt.FlowLayout(java.awt.FlowLayout.LEFT));

setelementmanagementLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
setelementmanagementLabel.setLabelFor(setelementmanagementComboBox);
setelementmanagementLabel.setText("Set Element Management:");
jPanel1.add(setelementmanagementLabel);

setelementmanagementComboBox.setPreferredSize(new java.awt.Dimension(175, 25));
setelementmanagementComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      setelementmanagementComboBoxActionPerformed(evt);
   }
});

jPanel1.add(setelementmanagementComboBox);

setelementdeleteButton.setText("Delete");
setelementdeleteButton.setToolTipText("Deletes the currently selected set element.");
setelementdeleteButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      setelementdeleteButtonActionPerformed(evt);
   }
});

jPanel1.add(setelementdeleteButton);

sdtToolBar.add(jPanel1);

getContentPane().add(sdtToolBar, java.awt.BorderLayout.NORTH);

feedbackareaScrollPane.setPreferredSize(new java.awt.Dimension(3, 100));
feedbackTextArea.setEditable(false);
feedbackTextArea.setText("Feedback info goes here...");
feedbackareaScrollPane.setViewportView(feedbackTextArea);

getContentPane().add(feedbackareaScrollPane, java.awt.BorderLayout.SOUTH);
```

```
treePanel.setLayout(new java.awt.GridLayout(2, 1));

libraryScrollPane.setPreferredSize(new java.awt.Dimension(175, 363));
libraryTree.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        libraryTreeMousePressed(evt);
    }
});
libraryTree.addTreeExpansionListener(new javax.swing.event.TreeExpansionListener() {
    public void treeCollapsed(javax.swing.event.TreeExpansionEvent evt) {
    }
    public void treeExpanded(javax.swing.event.TreeExpansionEvent evt) {
        libraryTreeTreeExpanded(evt);
    }
});

libraryScrollPane.setViewportView(libraryTree);

treePanel.add(libraryScrollPane);

scenarioTree.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        scenarioTreeMousePressed(evt);
    }
});

scenarioScrollPane.setViewportView(scenarioTree);

treePanel.add(scenarioScrollPane);

getContentPane().add(treePanel, java.awt.BorderLayout.WEST);

workareaTabbedPane.addChangeListener(new javax.swing.event.ChangeListener() {
    public void stateChanged(javax.swing.event.ChangeEvent evt) {
        workareaTabbedPaneStateChanged(evt);
    }
});
workareaTabbedPane.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mousePressed(java.awt.event.MouseEvent evt) {
        workareaTabbedPaneMousePressed(evt);
    }
});

getContentPane().add(workareaTabbedPane, java.awt.BorderLayout.CENTER);

fileMenu.setText("File");
newsubMenu.setText("New");
assetMenuItem.setText("Asset");
assetMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetMenuItemActionPerformed(evt);
    }
});

newsubMenu.add(assetMenuItem);

assetGoalMenuItem.setText("Goal");
assetGoalMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetGoalMenuItemActionPerformed(evt);
    }
});
```

```
newsubMenu.add(assetGoalMenuItem);

catalogcompMenuItem.setText("Catalog Component");
catalogcompMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      catalogcompMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(catalogcompMenuItem);

conditionMenuItem.setText("Condition");
conditionMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      conditionMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(conditionMenuItem);

dacGroupMenuItem.setText("DAC Group");
dacGroupMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      dacGroupMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(dacGroupMenuItem);

departmentMenuItem.setText("Department");
departmentMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      departmentMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(departmentMenuItem);

integrityMenuItem.setText("Integrity");
integrityMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      integrityMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(integrityMenuItem);

networkMenuItem.setText("Network");
networkMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      networkMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(networkMenuItem);

otherMenu.setText("Other");
filterMenuItem.setText("Filter");
filterMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      filterMenuItemActionPerformed(evt);
```

```
    }
});

otherMenu.add(filterMenuItem);

networkConnectionMenuItem.setText("Component Network Connection");
networkConnectionMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        networkConnectionMenuItemActionPerformed(evt);
    }
});

otherMenu.add(networkConnectionMenuItem);

proceduralSettingsMenuItem.setText("Procedural Settings");
proceduralSettingsMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        proceduralSettingsMenuItemActionPerformed(evt);
    }
});

otherMenu.add(proceduralSettingsMenuItem);

newsubMenu.add(otherMenu);

physcompMenuItem.setText("Physical Component");
physcompMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        physcompMenuItemActionPerformed(evt);
    }
});

newsubMenu.add(physcompMenuItem);

secrecyMenuItem.setText("Secrecy");
secrecyMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        secrecyMenuItemActionPerformed(evt);
    }
});

newsubMenu.add(secrecyMenuItem);

triggerMenuItem.setText("Trigger");
triggerMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        triggerMenuItemActionPerformed(evt);
    }
});

newsubMenu.add(triggerMenuItem);

userMenuItem.setText("User");
userMenuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        userMenuItemActionPerformed(evt);
    }
});

newsubMenu.add(userMenuItem);

workspaceMenuItem.setText("Workspace");
```

```
workspaceMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      workspaceMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(workspaceMenuItem);

zoneMenuItem.setText("Zone");
zoneMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      zoneMenuItemActionPerformed(evt);
   }
});

newsubMenu.add(zoneMenuItem);

fileMenu.add(newsubMenu);

openMenuItem.setText("Open...");
openMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      openMenuItemActionPerformed(evt);
   }
});

fileMenu.add(openMenuItem);

fileMenu.add(jSeparator1);

saveMenuItem.setText("Save");
saveMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      saveMenuItemActionPerformed(evt);
   }
});

fileMenu.add(saveMenuItem);

savescenarioMenuItem.setText("Save Scenario");
savescenarioMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      savescenarioMenuItemActionPerformed(evt);
   }
});

fileMenu.add(savescenarioMenuItem);

saveallMenuItem.setText("Save All");
saveallMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      saveallMenuItemActionPerformed(evt);
   }
});

fileMenu.add(saveallMenuItem);

saveasMenuItem.setText("Save As");
saveasMenuItem.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      saveasMenuItemActionPerformed(evt);
   }
```

```
        });

        fileMenu.add(saveasMenuItem);

        savescenarioasMenuItem.setText("Save Scenario As");
        savescenarioasMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                savescenarioasMenuItemActionPerformed(evt);
            }
        });

        fileMenu.add(savescenarioasMenuItem);

        fileMenu.add(jSeparator2);

        exitMenuItem.setText("Exit");
        exitMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                exitMenuItemActionPerformed(evt);
            }
        });

        fileMenu.add(exitMenuItem);

        sdtMenuBar.add(fileMenu);

        toolsMenu.setText("Tools");
        buildMenuItem.setText("Build");
        buildMenuItem.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                buildMenuItemActionPerformed(evt);
            }
        });

        toolsMenu.add(buildMenuItem);

        sdtMenuBar.add(toolsMenu);

        setJMenuBar(sdtMenuBar);

        java.awt.Dimension screenSize = java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-800)/2, (screenSize.height-600)/2, 800, 600);
    }
//networkConnectionMenuItemActionPerformed()
    //- creates a new network connection object
    //- initializes its dynamic content (based on what has been added to the
    // scenario)
    //- creates a new scenario element set
    //- prompts for a name for the new set
    //- tests the name provided to make sure it is legal
    //- appends the file extension for a network connection .KMS to the file
    //name
    //- adds the network connection object to the scenario element set
    //- creates a new tab in the tabbed work area, names it and adds the
    //new object to it
    //- if the file name provided was not legal the method prompts for
    //input until it is legal
    private void networkConnectionMenuItemActionPerformed(java.awt.event.ActionEvent evt)
    {
        NetworkConnection newNetworkConnection = new NetworkConnection();
        newNetworkConnection.populateDynamicSourceLists(datapath,startupScenario);
        ScenarioElementSet tempNetworkConnectionset = new ScenarioElementSet();
```

```
        tempNetworkConnectionset.setelementSetName(JOptionPane.showInputDialog(
                    "Enter the Component Network Connection file name."));

        if(tempNetworkConnectionset.getelementSetName() != null &&
          tempNetworkConnectionset.getelementSetName() != " " &&
          tempNetworkConnectionset.getelementSetName().length() != 0)
        {
           tempNetworkConnectionset.setelementSetName(tempNetworkConnectionset.
                                        getelementSetName()+".KMS");
           tempNetworkConnectionset.setelementType("Network Connection");
           tempNetworkConnectionset.elementContainer.add(newNetworkConnection);
           workareaTabbedPane.addTab("Component Network Connection:"+
                    tempNetworkConnectionset.getelementSetName(),
                    (NetworkConnection)tempNetworkConnectionset.
                    elementContainer.lastElement());
           tabManager.insertElementAt(tempNetworkConnectionset,
                               (workareaTabbedPane.getTabCount()-1));
        }
        else
        {
           JOptionPane.showMessageDialog(this,
                        "A file name is required",
                        "File Name Error", JOptionPane.ERROR_MESSAGE);
           networkConnectionMenuItemActionPerformed(null);
        }
     }
  //proceduralSettingsMenuItemActionPerformed()
     //- creates a new procedural settings object
     //- initializes its static content (based on what was read in by
     //readINIFiles()
     //- initializes its dynamic content (based on what has been added to the
     // scenario)
     //- creates a new scenario element set
     //- prompts for a name for the new set
     //- tests the name provided to make sure it is legal
     //- appends the file extension for a procedural settings .LMS to the file
     //name
     //- adds the procedural settings object to the scenario element set
     //- creates a new tab in the tabbed work area, names it and adds the
     //new object to it
     //- if the file name provided was not legal the method prompts for
     //input until it is legal
     private void proceduralSettingsMenuItemActionPerformed(java.awt.event.ActionEvent evt)
     {
        ProceduralSettings newProceduralSettings = new ProceduralSettings();
        newProceduralSettings.populateStaticSourceLists(
                fileContents_passcomplexity, fileContents_passlength,
                fileContents_passchangefreq);
        newProceduralSettings.populateDynamicSourceLists(datapath,startupScenario);
        ScenarioElementSet tempProceduralSettingsset = new ScenarioElementSet();
        tempProceduralSettingsset.setelementSetName(JOptionPane.showInputDialog(
                        "Enter the Procedural Settings file name."));

        if(tempProceduralSettingsset.getelementSetName() != null &&
          tempProceduralSettingsset.getelementSetName() != " " &&
          tempProceduralSettingsset.getelementSetName().length() != 0)
        {
        tempProceduralSettingsset.setelementSetName(tempProceduralSettingsset.
                                        getelementSetName()+".LMS");
           tempProceduralSettingsset.setelementType("Procedural Settings");
        tempProceduralSettingsset.elementContainer.add(newProceduralSettings);
           workareaTabbedPane.addTab("Procedural Settings:"+
```

```
                    tempProceduralSettingsset.getelementSetName(),
                    (ProceduralSettings)tempProceduralSettingsset.
                            elementContainer.lastElement());
        tabManager.insertElementAt(tempProceduralSettingsset,
                        (workareaTabbedPane.getTabCount()-1));
      }
      else
      {
        JOptionPane.showMessageDialog(this,
                    "A file name is required",
                    "File Name Error", JOptionPane.ERROR_MESSAGE);
        proceduralSettingsMenuItemActionPerformed(null);
      }
    }
  //filterMenuItemActionPerformed()
    //- creates a new filter object
    //- initializes its static content (based on what was read in by
    //readINIFiles()
    //- initializes its dynamic content (based on what has been added to the
    // scenario)
    //- creates a new scenario element set
    //- prompts for a name for the new set
    //- tests the name provided to make sure it is legal
    //- appends the file extension for a filter .FMS to the file
    //name
    //- adds the procedural settings object to the scenario element set
    //- creates a new tab in the tabbed work area, names it and adds the
    //new object to it
    //- if the file name provided was not legal the method prompts for
    //input until it is legal
    private void filterMenuItemActionPerformed(java.awt.event.ActionEvent evt)
    {
      Filter newFilter = new Filter();
      newFilter.populateDynamicSourceLists(datapath,startupScenario);
      newFilter.populateStaticSourceLists(fileContents_filterblocking,
                                fileContents_filterapps);
      ScenarioElementSet tempFilterset = new ScenarioElementSet();
      tempFilterset.setelementSetName(JOptionPane.showInputDialog(
                              "Enter the Filter file name."));

      if(tempFilterset.getelementSetName() != null &&
        tempFilterset.getelementSetName() != " " &&
        tempFilterset.getelementSetName().length() != 0)
      {
        tempFilterset.setelementSetName(tempFilterset.
                                getelementSetName()+".FMS");
        tempFilterset.setelementType("Filter");
        tempFilterset.elementContainer.add(newFilter);
        workareaTabbedPane.addTab("Filter:"+ tempFilterset.
                getelementSetName(),
                (Filter)tempFilterset.elementContainer.lastElement());
        tabManager.insertElementAt(tempFilterset,
                        (workareaTabbedPane.getTabCount()-1));
      }
      else
      {
        JOptionPane.showMessageDialog(this,
                    "A file name is required",
                    "File Name Error", JOptionPane.ERROR_MESSAGE);
        filterMenuItemActionPerformed(null);
      }
    }
```

```
//workspaceMenuItemActionPerformed()
   //- creates a new workspace object
   //- creates a new scenario element set
   //- prompts for a name for the new set
   //- tests the name provided to make sure it is legal
   //- appends the file extension for a workspace .WMS to the file
   //name
   //- adds the workspace object to the scenario element set
   //- creates a new tab in the tabbed work area, names it and adds the
   //new object to it
   //- if the file name provided was not legal the method prompts for
   //input until it is legal
   private void workspaceMenuItemActionPerformed(java.awt.event.ActionEvent evt)
   {
      Workspace newWorkspace = new Workspace();
      ScenarioElementSet tempWorkspaceset = new ScenarioElementSet();
      tempWorkspaceset.setelementSetName(JOptionPane.showInputDialog(
                              "Enter the Workspace file name."));
      if(tempWorkspaceset.getelementSetName() != null &&
        tempWorkspaceset.getelementSetName() != " " &&
        tempWorkspaceset.getelementSetName().length() != 0)
      {
         tempWorkspaceset.setelementSetName(tempWorkspaceset.
                                    getelementSetName()+".WMS");
         tempWorkspaceset.setelementType("Workspace");
         tempWorkspaceset.elementContainer.add(newWorkspace);
         workareaTabbedPane.addTab("Workspace:"+
                        tempWorkspaceset.getelementSetName(),
                        (Workspace)tempWorkspaceset.
                        elementContainer.lastElement());
         tabManager.insertElementAt(tempWorkspaceset,
                        (workareaTabbedPane.getTabCount()-1));
      }
      else
      {
         JOptionPane.showMessageDialog(this,
                     "A file name is required",
                     "File Name Error", JOptionPane.ERROR_MESSAGE);
         workspaceMenuItemActionPerformed(null);
      }
   }
//userMenuItemActionPerformed()
   //- creates a new user object
   //- initializes its dynamic content (based on what has been added to the
   // scenario)
   //- creates a new scenario element set
   //- prompts for a name for the new set
   //- tests the name provided to make sure it is legal
   //- appends the file extension for a user .UMS to the file
   //name
   //- adds the user object to the scenario element set
   //- creates a new tab in the tabbed work area, names it and adds the
   //new object to it
   //- if the file name provided was not legal the method prompts for
   //input until it is legal
   private void userMenuItemActionPerformed(java.awt.event.ActionEvent evt)
   {
      User newUser = new User();
      newUser.populateDynamicSourceLists(datapath,startupScenario);
      ScenarioElementSet tempUserset = new ScenarioElementSet();
      tempUserset.setelementSetName(JOptionPane.showInputDialog("Enter the User file name."));
      if(tempUserset.getelementSetName() != null &&
```

```
                tempUserset.getelementSetName() != " " &&
                tempUserset.getelementSetName().length() != 0)
            {
                tempUserset.setelementSetName(tempUserset. getelementSetName()+".UMS");
                tempUserset.setelementType("User");
                tempUserset.elementContainer.add(newUser);
                workareaTabbedPane.addTab("User:"+
tempUserset.getelementSetName(),(User)tempUserset.elementContainer.lastElement());
                tabManager.insertElementAt(tempUserset,(workareaTabbedPane.getTabCount()-1));
            }
            else
            {
                JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                userMenuItemActionPerformed(null);
            }
        }
    //triggerMenuItemActionPerformed()
        //- creates a new trigger object
        //- initializes its static content (based on what was read in by
        //readINIFiles()
        //- initializes its dynamic content (based on what has been added to the
        // scenario)
        //- creates a new scenario element set
        //- prompts for a name for the new set
        //- tests the name provided to make sure it is legal
        //- appends the file extension for a trigger .TMS to the file
        //name
        //- adds the trigger object to the scenario element set
        //- creates a new tab in the tabbed work area, names it and adds the
        //new object to it
        //- if the file name provided was not legal the method prompts for
        //input until it is legal
        private void triggerMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {
            Trigger newTrigger = new Trigger();
            newTrigger.populateStaticSourceLists(fileContents_trigger);
            newTrigger.populateDynamicSourceLists(datapath,startupScenario);
            ScenarioElementSet tempTriggerset = new ScenarioElementSet();
            tempTriggerset.setelementSetName(JOptionPane.showInputDialog("Enter the Trigger file name."));
            if(tempTriggerset.getelementSetName() != null && tempTriggerset.getelementSetName() != " " &&
tempTriggerset.getelementSetName().length() != 0)
            {
                tempTriggerset.setelementSetName(tempTriggerset.getelementSetName()+".TMS");
                tempTriggerset.setelementType("Trigger");
                tempTriggerset.elementContainer.add(newTrigger);
                workareaTabbedPane.addTab("Trigger:"+
tempTriggerset.getelementSetName(),(Trigger)tempTriggerset.elementContainer.lastElement());
                tabManager.insertElementAt(tempTriggerset, (workareaTabbedPane.getTabCount()-1));
            }
            else
            {
                JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                triggerMenuItemActionPerformed(null);
            }
        }
    //departmentMenuItemActionPerformed()
        //- creates a new department object
        //- initializes its static content (based on what was read in by
        //readINIFiles()
        //- initializes its dynamic content (based on what has been added to the
```

```
            // scenario)
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a department .EMS to the file
            //name
            //- adds the department object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void departmentMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               Department newDepartment = new Department();
               ScenarioElementSet tempDepartmentset = new ScenarioElementSet();
               tempDepartmentset.setelementSetName(JOptionPane.showInputDialog("Enter the Department file
name."));

               if(tempDepartmentset.getelementSetName() != null && tempDepartmentset.getelementSetName() != "
" && tempDepartmentset.getelementSetName().length() != 0)
               {
                  tempDepartmentset.setelementSetName(tempDepartmentset.getelementSetName()+".EMS");
                  tempDepartmentset.setelementType("Department");
                  tempDepartmentset.elementContainer.add(newDepartment);
                  workareaTabbedPane.addTab("Department:"+
tempDepartmentset.getelementSetName(),(Department)tempDepartmentset.elementContainer.lastElement());
               tabManager.insertElementAt(tempDepartmentset, (workareaTabbedPane.getTabCount()-1));
               }
               else
               {
                  JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                  departmentMenuItemActionPerformed(null);
               }
            }
         //departmentMenuItemActionPerformed()
            //- creates a new dac group object
            //- initializes its static content (based on what was read in by
            //readINIFiles()
            //- initializes its dynamic content (based on what has been added to the
            // scenario)
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a dac group .DMS to the file
            //name
            //- adds the dac group object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void dacGroupMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               DACGroup newDACGroup = new DACGroup();
               newDACGroup.populateStaticSourceLists(fileContents_backgroundcheck);
               ScenarioElementSet tempDACGroupset = new ScenarioElementSet();
               tempDACGroupset.setelementSetName(JOptionPane.showInputDialog("Enter the DACGroup file
name."));

               if(tempDACGroupset.getelementSetName() != null && tempDACGroupset.getelementSetName() != " "
&& tempDACGroupset.getelementSetName().length() != 0)
               {
```

```
                    tempDACGroupset.setelementSetName(tempDACGroupset.getelementSetName()+".DMS");
                    tempDACGroupset.setelementType("DAC Group");
                    tempDACGroupset.elementContainer.add(newDACGroup);
                    workareaTabbedPane.addTab("DAC Group:"+
tempDACGroupset.getelementSetName(),(DACGroup)tempDACGroupset.elementContainer.lastElement());
                    tabManager.insertElementAt(tempDACGroupset, (workareaTabbedPane.getTabCount()-1));
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                    dacGroupMenuItemActionPerformed(null);
                 }
              }
           //departmentMenuItemActionPerformed()
              //- creates a new condition object
              //- initializes its static content (based on what was read in by
              //readINIFiles()
              //- initializes its dynamic content (based on what has been added to the
              // scenario)
              //- creates a new scenario element set
              //- prompts for a name for the new set
              //- tests the name provided to make sure it is legal
              //- appends the file extension for a condition .BMS to the file
              //name
              //- adds the condition object to the scenario element set
              //- creates a new tab in the tabbed work area, names it and adds the
              //new object to it
              //- if the file name provided was not legal the method prompts for
              //input until it is legal
              private void conditionMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 Condition newCondition = new Condition();
                 newCondition.populateStaticSourceLists(fileContents_condition);
                 ScenarioElementSet tempConditionset = new ScenarioElementSet();
                 tempConditionset.setelementSetName(JOptionPane.showInputDialog("Enter the Condition file
name."));

                 if(tempConditionset.getelementSetName() != null && tempConditionset.getelementSetName() != " "
&& tempConditionset.getelementSetName().length() != 0)
                 {
                    tempConditionset.setelementSetName(tempConditionset.getelementSetName()+".BMS");
                    tempConditionset.setelementType("Condition");
                    tempConditionset.elementContainer.add(newCondition);

workareaTabbedPane.addTab("Condition:"+tempConditionset.getelementSetName(),(Condition)tempConditionset.ele
mentContainer.lastElement());
                    tabManager.insertElementAt(tempConditionset,(workareaTabbedPane.getTabCount()-1));
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                    conditionMenuItemActionPerformed(null);
                 }
              }
           //assetGoalMenuItemActionPerformed()
              //- creates a new asset goal object
              //- initializes its static content (based on what was read in by
              //readINIFiles()
              //- initializes its dynamic content (based on what has been added to the
              // scenario)
```

```
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a asset goal .GMS to the file
            //name
            //- adds the asset goal object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void assetGoalMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               AssetGoal newAssetGoal = new AssetGoal();
               newAssetGoal.populateDynamicSourceLists(datapath,startupScenario);
               ScenarioElementSet tempAssetGoalset = new ScenarioElementSet();
               tempAssetGoalset.setelementSetName(JOptionPane.showInputDialog("Enter the Goal file name."));

               if(tempAssetGoalset.getelementSetName() != null && tempAssetGoalset.getelementSetName() != " "
&& tempAssetGoalset.getelementSetName().length() != 0)
               {
                  tempAssetGoalset.setelementSetName(tempAssetGoalset.getelementSetName()+".GMS");
                  tempAssetGoalset.setelementType("Asset Goal");
                  tempAssetGoalset.elementContainer.add(newAssetGoal);

workareaTabbedPane.addTab("Goal:"+tempAssetGoalset.getelementSetName(),(AssetGoal)tempAssetGoalset.element
Container.lastElement());
                  tabManager.insertElementAt(tempAssetGoalset,(workareaTabbedPane.getTabCount()-1));
               }
               else
               {
                  JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                  assetGoalMenuItemActionPerformed(null);
               }
            }
         //assetMenuItemActionPerformed()
            //- creates a new asset object
            //- initializes its static content (based on what was read in by
            //readINIFiles()
            //- initializes its dynamic content (based on what has been added to the
            // scenario)
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a asset .AMS to the file
            //name
            //- adds the asset object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void assetMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               Asset newAsset = new Asset();
               newAsset.populateDynamicSourceLists(datapath,startupScenario);
               ScenarioElementSet tempAssetset = new ScenarioElementSet();
               tempAssetset.setelementSetName(JOptionPane.showInputDialog("Enter the Asset file name."));

               if(tempAssetset.getelementSetName() != null && tempAssetset.getelementSetName() != " " &&
tempAssetset.getelementSetName().length() != 0)
               {
                  tempAssetset.setelementSetName(tempAssetset.getelementSetName()+".AMS");
```

```
                    tempAssetset.setelementType("Asset");
                    tempAssetset.elementContainer.add(newAsset);

workareaTabbedPane.addTab("Asset:"+tempAssetset.getelementSetName(),(Asset)tempAssetset.elementContainer.last
Element());
                    tabManager.insertElementAt(tempAssetset,(workareaTabbedPane.getTabCount()-1));
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                    assetMenuItemActionPerformed(null);
                 }
              }
           //zoneMenuItemActionPerformed()
              //- creates a new zone object
              //- initializes its static content (based on what was read in by
              //readINIFiles()
              //- initializes its dynamic content (based on what has been added to the
              // scenario)
              //- creates a new scenario element set
              //- prompts for a name for the new set
              //- tests the name provided to make sure it is legal
              //- appends the file extension for a zone .ZMS to the file
              //name
              //- adds the zone object to the scenario element set
              //- creates a new tab in the tabbed work area, names it and adds the
              //new object to it
              //- if the file name provided was not legal the method prompts for
              //input until it is legal
              private void zoneMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 Zone newzone = new Zone();
                 newzone.populateDynamicSourceLists(datapath,startupScenario);
                 ScenarioElementSet tempZoneset = new ScenarioElementSet();
                 tempZoneset.setelementSetName(JOptionPane.showInputDialog("Enter the Zone file name."));

                 if(tempZoneset.getelementSetName() != null && tempZoneset.getelementSetName() != " " &&
tempZoneset.getelementSetName().length() != 0)
                 {
                    tempZoneset.setelementSetName(tempZoneset.getelementSetName()+".ZMS");
                    tempZoneset.setelementType("Zone");
                    tempZoneset.elementContainer.add(newzone);

workareaTabbedPane.addTab("Zone:"+tempZoneset.getelementSetName(),(Zone)tempZoneset.elementContainer.lastE
lement());
                    tabManager.insertElementAt(tempZoneset,(workareaTabbedPane.getTabCount()-1));
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                    zoneMenuItemActionPerformed(null);
                 }
              }
           //catalogCompMenuItemActionPerformed()
              //- creates a new catalog component object
              //- initializes its static content (based on what was read in by
              //readINIFiles()
              //- initializes its dynamic content (based on what has been added to the
              // scenario)
              //- creates a new scenario element set
```

```
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a catalog component .CMS to the file
            //name
            //- adds the catalog component object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void catalogcompMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               CatalogComponent newcatalogcomponent = new CatalogComponent();

newcatalogcomponent.populateStaticSourceLists(fileContents_basecomp,fileContents_os,fileContents_passcomplexity
,fileContents_passlength,fileContents_software);
               ScenarioElementSet tempCatalogComponentset = new ScenarioElementSet();
               tempCatalogComponentset.setelementSetName(JOptionPane.showInputDialog("Enter the catalog
component file name."));

               if(tempCatalogComponentset.getelementSetName() != null &&
tempCatalogComponentset.getelementSetName() != " " && tempCatalogComponentset.getelementSetName().length()
!= 0)
               {

tempCatalogComponentset.setelementSetName(tempCatalogComponentset.getelementSetName()+".CMS");
                  tempCatalogComponentset.setelementType("Catalog Component");
                  tempCatalogComponentset.elementContainer.add(newcatalogcomponent);
                  workareaTabbedPane.addTab("Catalog
Component:"+tempCatalogComponentset.getelementSetName(),(CatalogComponent)tempCatalogComponentset.eleme
ntContainer.lastElement());
                  tabManager.insertElementAt(tempCatalogComponentset,(workareaTabbedPane.getTabCount()-1));
               }
               else
               {
                  JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                  catalogcompMenuItemActionPerformed(null);
               }
            }
         //physCompMenuItemActionPerformed()
            //- creates a new physical component object
            //- initializes its static content (based on what was read in by
            //readINIFiles()
            //- initializes its dynamic content (based on what has been added to the
            // scenario)
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a physical component .PMS to the file
            //name
            //- adds the physicla component object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void physcompMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               PhysicalComponent newphysicalcomponent = new PhysicalComponent();

newphysicalcomponent.populateStaticSourceLists(fileContents_basecomp,fileContents_os,fileContents_passcomplexit
y,fileContents_passlength,fileContents_software);
               newphysicalcomponent.populateDynamicSourceLists(datapath,startupScenario);
```

```
                ScenarioElementSet tempPhysicalComponentset = new ScenarioElementSet();
                tempPhysicalComponentset.setelementSetName(JOptionPane.showInputDialog("Enter the physical
component file name."));

                if(tempPhysicalComponentset.getelementSetName() != null &&
tempPhysicalComponentset.getelementSetName() != " " &&
tempPhysicalComponentset.getelementSetName().length() != 0)
                {

tempPhysicalComponentset.setelementSetName(tempPhysicalComponentset.getelementSetName()+".PMS");
                   tempPhysicalComponentset.setelementType("Physical Component");
                   tempPhysicalComponentset.elementContainer.add(newphysicalcomponent);
                   workareaTabbedPane.addTab("Physical
Component:"+tempPhysicalComponentset.getelementSetName(),(PhysicalComponent)tempPhysicalComponentset.ele
mentContainer.lastElement());
                   tabManager.insertElementAt(tempPhysicalComponentset,(workareaTabbedPane.getTabCount()-1));
                }
                else
                {
                   JOptionPane.showMessageDialog(this,"A file name is required","File Name Error",
JOptionPane.ERROR_MESSAGE);
                   physcompMenuItemActionPerformed(null);
                }
             }
          //waRemoveMenuItemActionPerformed()
             //closes the currently selected tab when the remove option is chosen
             //from the right click menu for a tab
             private void waRemoveMenuItemActionPerformed(java.awt.event.ActionEvent evt)
             {
             int targetTab = -1;
                targetTab = workareaTabbedPane.getSelectedIndex();
                if(targetTab != 0)//do not allow the scenario tab to be removed
                {
                   workareaTabbedPane.remove(targetTab);
                   tabManager.remove(targetTab);
                }
             }
          //waAddMenuItemActionPerformed()
             //for each descriptor - except scenario - which cannot be added to
             //itself - the following action are performed
             //- based on the descriptor type as provided by the currently selected tab
             //- get the appropriate vector from the scenario manager and store it as
              //a generic object
             //- cast the generic object to a vector
             //- get the object to be added to the scenario from the tab manager
             //- cast the generic object as a scenario element set
             //- add the scenario elelment set name (filename) to the scenario
             //NOTE: filenames are used to keep the scenario dynamic.  In an earlier
             //version copies of the actual objects were stored in the scenario manager
             //and reusability failed.
             //- update the scenario tree
             //- save the scenario element set whose name was just added to the
             //scenario manager
             private void waAddMenuItemActionPerformed(java.awt.event.ActionEvent evt)
             {
                Object tempObject = new Object();
                Object tempObject1 = new Object();
                ScenarioElementSet tempSet = new ScenarioElementSet();
                java.util.Vector tempVec = new java.util.Vector();
                java.util.Vector tempVec1 = new java.util.Vector();
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
                {
```

```
                    tempObject = startupScenario.scenarioManager.get(0);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());//save the filename
                    //this will make the scenario manager dynamic
                    initScenarioTree();//update the tree
                    saveMenuItemActionPerformed(null);//save the set first
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                {
                    tempObject = startupScenario.scenarioManager.get(1);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                {
                    tempObject = startupScenario.scenarioManager.get(2);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
                {
                    tempObject = startupScenario.scenarioManager.get(3);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                {
                    tempObject = startupScenario.scenarioManager.get(4);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                {
                    tempObject = startupScenario.scenarioManager.get(5);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    saveMenuItemActionPerformed(null);
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
```

```
                {
                   tempObject = startupScenario.scenarioManager.get(6);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                {
                   tempObject = startupScenario.scenarioManager.get(7);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                {
                   tempObject = startupScenario.scenarioManager.get(8);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                {
                   tempObject = startupScenario.scenarioManager.get(9);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                {
                   tempObject = startupScenario.scenarioManager.get(10);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
                {
                   tempObject = startupScenario.scenarioManager.get(11);
                   tempVec = (java.util.Vector)tempObject;
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempVec.addElement(tempSet.getelementSetName());
                   initScenarioTree();
                   saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Workspace"))
                {
```

```
                    tempObject = startupScenario.scenarioManager.get(12);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                {
                    tempObject = startupScenario.scenarioManager.get(13);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(0);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural
Settings"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(1);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component
Network Connection"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(2);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                    saveMenuItemActionPerformed(null);
                }
            }
        //stRemoveMenuItem
            //for each descriptor - except scenario - which cannot be removed from
            //itself - the following action are performed
            //- based on the descriptor extension type as provided by the
            //tree path in scenario tree
```

```
//- using regular expressions convert the tree path to a usable form
//this will give you the filename to be deleted.
//- get the appropriate vector from the scenario manager and store it as
//a generic object
//- cast the generic object to a vector
//- for each object (set name )in the vector check its name against the name
//obtained from the tree path
//- if there is a match remove the set from the scenario
private void stRemoveMenuItemActionPerformed(java.awt.event.ActionEvent evt)
{
   Object tempObject = new Object();
   Object tempObject1 = new Object();
    String tempSetName = new String();
   String selPath = new String();
   java.util.Vector tempVec = new java.util.Vector();
   java.util.Vector tempVec1 = new java.util.Vector();
   selPath = rightClickTreePath;
   //asset
   if(selPath.toString().endsWith(".AMS]"))
   {
      String tempStr = selPath.toString();
      tempStr = tempStr.replaceAll("\\[.*sset, "," ");
      tempStr = tempStr.replaceAll("]"," ");
      tempStr = tempStr.trim();
      tempObject = startupScenario.scenarioManager.get(0);
      tempVec = (java.util.Vector)tempObject;
      for(int i=0; i<tempVec.size(); i++)
      {
         tempObject = tempVec.get(i);
         tempSetName = (String)tempObject;
         if(tempSetName.equalsIgnoreCase(tempStr))
         {
            tempVec.remove(i);
            break;
         }
      }
      initScenarioTree();
   }
   //asset goal
   if(selPath.toString().endsWith(".GMS]"))
   {
      String tempStr = selPath.toString();
      tempStr = tempStr.replaceAll("\\[.*oal, "," ");
      tempStr = tempStr.replaceAll("]"," ");
      tempStr = tempStr.trim();
      tempObject = startupScenario.scenarioManager.get(1);
      tempVec = (java.util.Vector)tempObject;
      for(int i=0; i<tempVec.size(); i++)
      {
         tempObject = tempVec.get(i);
         tempSetName = (String)tempObject;
         if(tempSetName.equalsIgnoreCase(tempStr))
         {
            tempVec.remove(i);
            break;
         }
      }
      initScenarioTree();
   }
   //catalog component
   if(selPath.toString().endsWith(".CMS]"))
   {
```

```
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ent, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(2);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//condition
if(selPath.toString().endsWith(".BMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ion, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(3);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//dac group
if(selPath.toString().endsWith(".DMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*oup, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(4);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//departemnt
if(selPath.toString().endsWith(".EMS]"))
```

```
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ent, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(5);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//integrity
if(selPath.toString().endsWith(".IMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*y, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(6);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//netowrk
if(selPath.toString().endsWith(".NMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*k, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(7);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//physical component
```

```
if(selPath.toString().endsWith(".PMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ent, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(8);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//secrecy
if(selPath.toString().endsWith(".SMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*y, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   feedbackTextArea.append(tempStr+"\n");
   tempObject = startupScenario.scenarioManager.get(9);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//trigger
if(selPath.toString().endsWith(".TMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ger, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(10);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
```

```
}
//user
if(selPath.toString().endsWith(".UMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ser, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(11);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//workspace
if(selPath.toString().endsWith(".WMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*ace, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(12);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
   initScenarioTree();
}
//zone
if(selPath.toString().endsWith(".ZMS]"))
{
   String tempStr = selPath.toString();
   tempStr = tempStr.replaceAll("\\[.*one, "," ");
   tempStr = tempStr.replaceAll("]"," ");
   tempStr = tempStr.trim();
   tempObject = startupScenario.scenarioManager.get(13);
   tempVec = (java.util.Vector)tempObject;
   for(int i=0; i<tempVec.size(); i++)
   {
      tempObject = tempVec.get(i);
      tempSetName = (String)tempObject;
      if(tempSetName.equalsIgnoreCase(tempStr))
      {
         tempVec.remove(i);
         break;
      }
   }
```

```
    initScenarioTree();
}
//filter
if(selPath.toString().endsWith(".FMS]"))
{
    String tempStr = selPath.toString();
    tempStr = tempStr.replaceAll("\\[.*ter, "," ");
    tempStr = tempStr.replaceAll("]"," ");
    tempStr = tempStr.trim();
    feedbackTextArea.append("this is what tempstr is: " + tempStr+"\n");
    tempObject = startupScenario.scenarioManager.get(14);
    tempVec = (java.util.Vector)tempObject;
    tempObject1 = tempVec.get(0);
    tempVec1 = (java.util.Vector)tempObject1;
    for(int i=0; i<tempVec1.size(); i++)
    {
        tempObject1 = tempVec1.get(i);
        tempSetName = (String)tempObject1;
        if(tempSetName.equalsIgnoreCase(tempStr))
        {
            tempVec1.remove(i);
            break;
        }

    }
    initScenarioTree();
}
//procedural
if(selPath.toString().endsWith(".LMS]"))
{
    String tempStr = selPath.toString();
    tempStr = tempStr.replaceAll("\\[.*ngs, "," ");
    tempStr = tempStr.replaceAll("]"," ");
    tempStr = tempStr.trim();
    tempObject = startupScenario.scenarioManager.get(14);
    tempVec = (java.util.Vector)tempObject;
    tempObject1 = tempVec.get(1);
    tempVec1 = (java.util.Vector)tempObject1;
    for(int i=0; i<tempVec1.size(); i++)
    {
        tempObject1 = tempVec1.get(i);
        tempSetName = (String)tempObject1;
        if(tempSetName.equalsIgnoreCase(tempStr))
        {
            tempVec1.remove(i);
            break;
        }
    }
    initScenarioTree();
}
//network connection
if(selPath.toString().endsWith(".KMS]"))
{
    String tempStr = selPath.toString();
    tempStr = tempStr.replaceAll("\\[.*tion, "," ");
    tempStr = tempStr.replaceAll("]"," ");
    tempStr = tempStr.trim();
    tempObject = startupScenario.scenarioManager.get(14);
    tempVec = (java.util.Vector)tempObject;
    tempObject1 = tempVec.get(2);
    tempVec1 = (java.util.Vector)tempObject1;
    for(int i=0; i<tempVec1.size(); i++)
```

```
                    {
                       tempObject1 = tempVec1.get(i);
                       tempSetName = (String)tempObject1;
                       if(tempSetName.equalsIgnoreCase(tempStr))
                       {
                          tempVec1.remove(i);
                          break;
                       }
                    }
                    initScenarioTree();
                 }
              }
           //stOpenMenuItemActionPerformed
              //does the exact same thing that ltOpenMenuItemActionPerformed does
              //opens the set that was right clicked on
              private void stOpenMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 ltOpenMenuItemActionPerformed(null);
              }
           //ltDeleteMenuItemActionPerformed
              //permanently removes the file that was right clicked on in the
              //library tree from the file system
              //- derives the file to be removed by passing the tree path to the method
              //treePathToFilePath
              private void ltDeleteMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 String selPath = new String();
                 selPath = rightClickTreePath;
                 File targetFile = new File(treePathtoFilePath(selPath.toString()));
                 if(!targetFile.isDirectory())
                 {
                    if(targetFile.delete())
                    {
                       JOptionPane.showMessageDialog(this, "The file " + treePathtoFilePath(selPath.toString()) + " has
been deleted.", "File Deletion", JOptionPane.INFORMATION_MESSAGE);
                    }
                    initLibraryTree();
                 }
              }
           //ltAddMenuItemActionPerformed
              //adds the file slected by a right click on the library tree to the
              //scenario
              //- if the file chosen is not a directory
              //it prefectches the object based on the output of treePathtoFilePath
              //- then based on the path to the descriptors directory
              //- get the appropriate vector from the scenario manager and store it as
               //a generic object
              //- cast the generic object to a vector
              //- get the object to be added to the scenario from the tab manager
              //- cast the generic object as a scenario element set
              //- add the scenario elelment set name (filename) to the scenario
              //NOTE: filenames are used to keep the scenario dynamic.  In an earlier
              //version copies of the actual objects were stored in the scenario manager
              //and reusability failed.
              //- update the scenario tree
              //- save the scenario element set whose name was just added to the
              //scenario manager
              private void ltaddMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 String selPath = new String();
                 ScenarioElementSet tempSet = new ScenarioElementSet();
                 Object tempObject = new Object();
```

```
                Object tempObject1 = new Object();
                java.util.Vector tempVec = new java.util.Vector();
                java.util.Vector tempVec1 = new java.util.Vector();
                selPath = rightClickTreePath;
                File targetFile = new File(treePathtoFilePath(selPath.toString()));
                if(!targetFile.isDirectory())
                {
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(treePathtoFilePath(selPath.toString())));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                   }
                   if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset"))
                   {
                      tempObject = startupScenario.scenarioManager.get(0);
                      tempVec = (java.util.Vector)tempObject;
                      tempVec.addElement(tempSet.getelementSetName());
                      initScenarioTree();
                   }
                   if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal"))
                   {
                      tempObject = startupScenario.scenarioManager.get(1);
                      tempVec = (java.util.Vector)tempObject;
                      tempVec.addElement(tempSet.getelementSetName());
                      initScenarioTree();
                   }
                   if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog Component"))
                   {
                      tempObject = startupScenario.scenarioManager.get(2);
                      tempVec = (java.util.Vector)tempObject;
                      tempVec.addElement(tempSet.getelementSetName());
```

```
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition"))
                {
                    tempObject = startupScenario.scenarioManager.get(3);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group"))
                {
                    tempObject = startupScenario.scenarioManager.get(4);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department"))
                {
                    tempObject = startupScenario.scenarioManager.get(5);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity"))
                {
                    tempObject = startupScenario.scenarioManager.get(6);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network"))
                {
                    tempObject = startupScenario.scenarioManager.get(7);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component"))
                {
                    tempObject = startupScenario.scenarioManager.get(8);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy"))
                {
                    tempObject = startupScenario.scenarioManager.get(9);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger"))
                {
                    tempObject = startupScenario.scenarioManager.get(10);
```

```
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User"))
                {
                    tempObject = startupScenario.scenarioManager.get(11);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace"))
                {
                    tempObject = startupScenario.scenarioManager.get(12);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone"))
                {
                    tempObject = startupScenario.scenarioManager.get(13);
                    tempVec = (java.util.Vector)tempObject;
                    tempVec.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(0);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(1);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection"))
                {
                    tempObject = startupScenario.scenarioManager.get(14);
                    tempVec = (java.util.Vector)tempObject;
                    tempObject1 = tempVec.get(2);
                    tempVec1 = (java.util.Vector)tempObject1;
                    tempVec1.addElement(tempSet.getelementSetName());
                    initScenarioTree();
                }
            }
        }
    //ltOpenMenuItemActionPerformed
        //opens the file slected by a right click on the library tree \
```

```
            //prefectches the object based on the output of treePathtoFilePath
            //- then based on the path to the descriptors directory
            //- create a new tab with the appropriate descriptor title
            //- add the object from the file to the tab manager at a position
            //equal to the tab count -1 (tab count starts at 1 and the tab manager
            //starts at 0 so we always have to subtract 1).  By adding the set
            //to the tab manager at a position equal to the tab count - 1 the
            //two datastructures are always working in parallel to eachother.
            //- get the last element used from the from the set and reinitialize
            //its listeners
            private void ltOpenMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               String selPath = new String();
               selPath = rightClickTreePath;
               ScenarioElementSet tempSet = new ScenarioElementSet();
               Object tempObject = new Object();
               Workspace tempWorkspace = new Workspace();
               try
               {
                  input = new java.io.ObjectInputStream(new
java.io.FileInputStream(treePathtoFilePath(selPath.toString())));
               }
               catch(IOException ioException)
               {
                  JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
               }
               try
               {
                  tempSet = (ScenarioElementSet)input.readObject();
               }
               catch(ClassNotFoundException classnotfoundException)
               {
                  JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
               }
               catch(IOException ioException)
               {
                  JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
               }
               try
               {
                  input.close();
               }
               catch(IOException ioException)
               {
                  JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
               }
               //asset
               if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset"))
               {

workareaTabbedPane.addTab("Asset:"+tempSet.getelementSetName(),(Asset)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                  tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                  tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                  Asset tempAsset = new Asset();
                  for(int i = 0; i < tempSet.elementContainer.size(); i++)
                  {
```

```
                        tempObject = tempSet.elementContainer.get(i);
                        tempAsset = (Asset)tempObject;
                        tempAsset.reInitListeners(datapath,startupScenario);
                        tempAsset.reInitLists();
                        tempAsset.reInitButtons();
                     }
                  }
                  //asset goal
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal"))
                  {

workareaTabbedPane.addTab("Goal:"+tempSet.getelementSetName(),(AssetGoal)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     AssetGoal tempAssetGoal = new AssetGoal();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempAssetGoal = (AssetGoal)tempObject;
                        tempAssetGoal.reInitListeners(datapath,startupScenario);
                        tempAssetGoal.reInitLists();
                        tempAssetGoal.reInitButtons();
                     }
                  }
                  //catalog component
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog Component"))
                  {
                     workareaTabbedPane.addTab("Catalog
Component:"+tempSet.getelementSetName(),(CatalogComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     CatalogComponent tempCatalogComponent = new CatalogComponent();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempCatalogComponent = (CatalogComponent)tempObject;

tempCatalogComponent.reInitListeners(fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileConte
nts_passlength,fileContents_software);
                        tempCatalogComponent.reInitLists(fileContents_software);
                     }
                  }
                  //condition
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition"))
                  {

workareaTabbedPane.addTab("Condition:"+tempSet.getelementSetName(),(Condition)tempSet.elementContainer.elem
entAt(tempSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     Condition tempCondition = new Condition();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempCondition = (Condition)tempObject;
                        tempCondition.reInitListeners(fileContents_condition);
                     }
```

```
                }
                //dac group
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group"))
                {
                   workareaTabbedPane.addTab("DAC
Group:"+tempSet.getelementSetName(),(DACGroup)tempSet.elementContainer.elementAt(tempSet.getlastElementUs
ed()));
                   tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                }
                //department
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department"))
                {

workareaTabbedPane.addTab("Department:"+tempSet.getelementSetName(),(Department)tempSet.elementContainer.e
lementAt(tempSet.getlastElementUsed()));
                   tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                }
                //integrity
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity"))
                {

workareaTabbedPane.addTab("Integrity:"+tempSet.getelementSetName(),(Integrity)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                   tabManager.insertElementAt(tempSet,(workareaTabbedPane.getTabCount()-1));
                }
                //network
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network"))
                {

workareaTabbedPane.addTab("Network:"+tempSet.getelementSetName(),(Network)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                   tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                }
                //physical component
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component"))
                {
                   workareaTabbedPane.addTab("Physical
Component:"+tempSet.getelementSetName(),(PhysicalComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                   tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                   tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                   PhysicalComponent tempPhys = new PhysicalComponent();
                   for(int i = 0; i < tempSet.elementContainer.size(); i++)
                   {
                      tempObject = tempSet.elementContainer.get(i);
                      tempPhys = (PhysicalComponent)tempObject;

tempPhys.reInitListeners(datapath,startupScenario,fileContents_basecomp,fileContents_os,fileContents_passcomplexit
y,fileContents_passlength,fileContents_software);
                      tempPhys.reInitLists(fileContents_software);
                      tempPhys.reInitButtons();
                   }
                }
                //secrecy
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy"))
                {
```

```
workareaTabbedPane.addTab("Secrecy:"+tempSet.getelementSetName(),(Secrecy)tempSet.elementContainer.element
At(tempSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                }
                //tirigger
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger"))
                {

workareaTabbedPane.addTab("Trigger:"+tempSet.getelementSetName(),(Trigger)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    Trigger tempTrigger = new Trigger();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempTrigger = (Trigger)tempObject;
                       tempTrigger.reInitListeners(datapath,startupScenario,fileContents_trigger);
                    }
                }
                //user
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User"))
                {

workareaTabbedPane.addTab("User:"+tempSet.getelementSetName(),(User)tempSet.elementContainer.elementAt(tem
pSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    User tempUser = new User();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempUser = (User)tempObject;
                       tempUser.reInitListeners(datapath,startupScenario);
                       tempUser.reInitLists();
                       tempUser.reInitButtons();
                    }
                }
                //workspace
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace"))
                {

workareaTabbedPane.addTab("Workspace:"+tempSet.getelementSetName(),(Workspace)tempSet.elementContainer.el
ementAt(tempSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempWorkspace = (Workspace)tempObject;
                       tempWorkspace.reInitListeners();
                       tempWorkspace.reInitLists();
                    }
                }
                //filter
                if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter"))
                {
```

```
workareaTabbedPane.addTab("Filter:"+tempSet.getelementSetName(),(Filter)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    Filter tempFilter = new Filter();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempFilter = (Filter)tempObject;

tempFilter.reInitListeners(datapath,startupScenario,fileContents_filterblocking,fileContents_filterapps);
                       tempFilter.reInitLists(fileContents_filterapps);
                       tempFilter.reInitButtons();
                    }
                 }
                 //procedural settings
                 if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings"))
                 {
                    workareaTabbedPane.addTab("Procedural
Settings:"+tempSet.getelementSetName(),(ProceduralSettings)tempSet.elementContainer.elementAt(tempSet.getlastEl
ementUsed()));
                    tabManager.insertElementAt(tempSet,(workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempProceduralSettings = (ProceduralSettings)tempObject;

tempProceduralSettings.reInitListeners(datapath,startupScenario,fileContents_passcomplexity,fileContents_passlength,
fileContents_passchangefreq);
                       tempProceduralSettings.reInitLists();
                       tempProceduralSettings.reInitButtons();
                    }
                 }
                 //zone
                 if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone"))
                 {

workareaTabbedPane.addTab("Zone:"+tempSet.getelementSetName(),(Zone)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    Zone tempZone = new Zone();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempZone = (Zone)tempObject;
                       tempZone.reInitListeners(datapath,startupScenario);
                       tempZone.reInitLists();
                       tempZone.reInitButtons();
                    }
                 }
                 //network connection
                 if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection"))
                 {
```

```
                    workareaTabbedPane.addTab("Component Network
Connection:"+tempSet.getelementSetName(),(NetworkConnection)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                    tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                    NetworkConnection tempNetworkConnection = new NetworkConnection();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempNetworkConnection = (NetworkConnection)tempObject;
                       tempNetworkConnection.reInitListeners(datapath,startupScenario);
                       tempNetworkConnection.reInitLists();
                       tempNetworkConnection.reInitButtons();
                    }
                 }
              }
           //workareaTabbedPaneMousePressed()
              //the mouse event listener for the tabbed work area
              private void workareaTabbedPaneMousePressed(java.awt.event.MouseEvent evt)
              {
                 if(evt.getButton() == evt.BUTTON3)//right click
                 {
                    workAreaTabPopupMenu.show(evt.getComponent(),evt.getX(),evt.getY());
                 }
              }
           //scenarioTreeMousePressed()
              //the mouse event listener for the scenario tree.
              //on a double click it opens the file double clicked
              //on a right click it launches the right mouse button
              //pop up menu.
              //- prefectches the object based on the output of treePathtoFilePath
              //- then based on the path to the descriptors directory
              //- create a new tab with the appropriate descriptor title
              //- add the object from the file to the tab manager at a position
              //equal to the tab count -1 (tab count starts at 1 and the tab manager
              //starts at 0 so we always have to subtract 1).  By adding the set
              //to the tab manager at a position equal to the tab count - 1 the
              //two datastructures are always working in parallel to eachother.
              //- get the last element used from the from the set and reinitialize
              //its listeners
              private void scenarioTreeMousePressed(java.awt.event.MouseEvent evt)
              {
                 int selRow = scenarioTree.getRowForLocation(evt.getX(), evt.getY());
                 TreePath selPath = scenarioTree.getPathForLocation(evt.getX(),evt.getY());
                 Object tempObject = new Object();
                 Workspace tempWorkspace = new Workspace();
                 ScenarioElementSet tempSet = new ScenarioElementSet();
                 if(selRow != -1)
                 {//double click
                    if(evt.getButton() == evt.BUTTON1 && evt.getClickCount() == 2)
                    {
                       try
                       {
                          input = new java.io.ObjectInputStream(new
java.io.FileInputStream(treePathtoFilePath(selPath.toString())));
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
                       try
```

```
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    //asset
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset"))
                    {

workareaTabbedPane.addTab("Asset:"+tempSet.getelementSetName(),(Asset)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       Asset tempAsset = new Asset();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempAsset = (Asset)tempObject;
                          tempAsset.reInitListeners(datapath,startupScenario);
                          tempAsset.reInitLists();
                          tempAsset.reInitButtons();
                       }
                    }
                    //asset goal
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal"))
                    {

workareaTabbedPane.addTab("Goal:"+tempSet.getelementSetName(),(AssetGoal)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       AssetGoal tempAssetGoal = new AssetGoal();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempAssetGoal = (AssetGoal)tempObject;
                          tempAssetGoal.reInitListeners(datapath,startupScenario);
                          tempAssetGoal.reInitLists();
                          tempAssetGoal.reInitButtons();
                       }
                    }
                    //catalog component
```

```
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/CatalogComponent"))
                    {
                       workareaTabbedPane.addTab("Catalog
Component:"+tempSet.getelementSetName(),(CatalogComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        CatalogComponent tempCatalogComponent = new CatalogComponent();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempCatalogComponent = (CatalogComponent)tempObject;

tempCatalogComponent.reInitListeners(fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileConte
nts_passlength,fileContents_software);
                           tempCatalogComponent.reInitLists(fileContents_software);
                        }
                    }
                    //condition
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition"))
                    {

workareaTabbedPane.addTab("Condition:"+tempSet.getelementSetName(),(Condition)tempSet.elementContainer.elem
entAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Condition tempCondition = new Condition();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempCondition = (Condition)tempObject;
                           tempCondition.reInitListeners(fileContents_condition);
                        }
                    }
                    //dac group
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group"))
                    {
                       workareaTabbedPane.addTab("DAC
Group:"+tempSet.getelementSetName(),(DACGroup)tempSet.elementContainer.elementAt(tempSet.getlastElementUs
ed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //department
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department"))
                    {

workareaTabbedPane.addTab("Department:"+tempSet.getelementSetName(),(Department)tempSet.elementContainer.e
lementAt(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //integrity
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity"))
                    {

workareaTabbedPane.addTab("Integrity:"+tempSet.getelementSetName(),(Integrity)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet,(workareaTabbedPane.getTabCount()-1));
```

```
                    }
                    //network
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network"))
                    {

workareaTabbedPane.addTab("Network:"+tempSet.getelementSetName(),(Network)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //physical component
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component"))
                    {
                       workareaTabbedPane.addTab("Physical
Component:"+tempSet.getelementSetName(),(PhysicalComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       PhysicalComponent tempPhys = new PhysicalComponent();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempPhys = (PhysicalComponent)tempObject;

tempPhys.reInitListeners(datapath,startupScenario,fileContents_basecomp,fileContents_os,fileContents_passcomplexit
y,fileContents_passlength,fileContents_software);
                          tempPhys.reInitLists(fileContents_software);
                          tempPhys.reInitButtons();
                       }
                    }
                    //secrecy
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy"))
                    {

workareaTabbedPane.addTab("Secrecy:"+tempSet.getelementSetName(),(Secrecy)tempSet.elementContainer.element
At(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //trigger
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger"))
                    {

workareaTabbedPane.addTab("Trigger:"+tempSet.getelementSetName(),(Trigger)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       Trigger tempTrigger = new Trigger();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempTrigger = (Trigger)tempObject;
                          tempTrigger.reInitListeners(datapath,startupScenario,fileContents_trigger);
                       }
                    }
                    //user
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User"))
                    {
```

```
workareaTabbedPane.addTab("User:"+tempSet.getelementSetName(),(User)tempSet.elementContainer.elementAt(tem
pSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        User tempUser = new User();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempUser = (User)tempObject;
                           tempUser.reInitListeners(datapath,startupScenario);
                           tempUser.reInitLists();
                           tempUser.reInitButtons();
                        }
                     }
                     //workspace
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace"))
                     {

workareaTabbedPane.addTab("Workspace:"+tempSet.getelementSetName(),(Workspace)tempSet.elementContainer.el
ementAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempWorkspace = (Workspace)tempObject;
                           tempWorkspace.reInitListeners();
                           tempWorkspace.reInitLists();
                        }
                     }
                     //filter
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter"))
                     {

workareaTabbedPane.addTab("Filter:"+tempSet.getelementSetName(),(Filter)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Filter tempFilter = new Filter();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempFilter = (Filter)tempObject;

tempFilter.reInitListeners(datapath,startupScenario,fileContents_filterblocking,fileContents_filterapps);
                           tempFilter.reInitLists(fileContents_filterapps);
                           tempFilter.reInitButtons();
                        }
                     }
                     //procedural settings
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings"))
                     {
                        workareaTabbedPane.addTab("Procedural
Settings:"+tempSet.getelementSetName(),(ProceduralSettings)tempSet.elementContainer.elementAt(tempSet.getlastEl
ementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
```

```
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempProceduralSettings = (ProceduralSettings)tempObject;
                           tempProceduralSettings.reInitListeners(datapath,startupScenario,fileContents_passcomplexity,
fileContents_passlength, fileContents_passchangefreq);
                           tempProceduralSettings.reInitLists();
                           tempProceduralSettings.reInitButtons();
                        }
                     }
                     //zone
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone"))
                     {

workareaTabbedPane.addTab("Zone:"+tempSet.getelementSetName(),(Zone)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Zone tempZone = new Zone();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempZone = (Zone)tempObject;
                           tempZone.reInitListeners(datapath,startupScenario);
                           tempZone.reInitLists();
                           tempZone.reInitButtons();
                        }
                     }
                     //network connection
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection"))
                     {
                        workareaTabbedPane.addTab("Component Network
Connection:"+tempSet.getelementSetName(),(NetworkConnection)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        NetworkConnection tempNetworkConnection = new NetworkConnection();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempNetworkConnection = (NetworkConnection)tempObject;
                           tempNetworkConnection.reInitListeners(datapath,startupScenario);
                           tempNetworkConnection.reInitLists();
                           tempNetworkConnection.reInitButtons();
                        }
                     }
                  }//right click
                  else if(evt.getButton() == evt.BUTTON3 && evt.getClickCount() == 1)
                  {
                     rightClickTreePath = selPath.toString();
                     scenarioTreePopupMenu.show(evt.getComponent(), evt.getX(), evt.getY());
                  }
               }
            }
         //libraryTreeMousePressed()
            //the mouse event listener for the library tree.
            //on a double click it opens the file double clicked
            //on a right click it launches the right mouse button
            //pop up menu.
            //- prefectches the object based on the output of treePathtoFilePath
            //- then based on the path to the descriptors directory
```

```
            //- create a new tab with the appropriate descriptor title
            //- add the object from the file to the tab manager at a position
            //equal to the tab count -1 (tab count starts at 1 and the tab manager
            //starts at 0 so we always have to subtract 1).  By adding the set
            //to the tab manager at a position equal to the tab count - 1 the
            //two datastructures are always working in parallel to eachother.
            //- get the last element used from the from the set and reinitialize
            //its listeners
            private void libraryTreeMousePressed(java.awt.event.MouseEvent evt)
            {
               int selRow = libraryTree.getRowForLocation(evt.getX(), evt.getY());
               TreePath selPath = libraryTree.getPathForLocation(evt.getX(),evt.getY());
               ScenarioElementSet tempSet = new ScenarioElementSet();
               Object tempObject = new Object();
               if(selRow != -1)
               {//double click
                  if(evt.getButton() == evt.BUTTON1 && evt.getClickCount() == 2)
                  {
                     try
                     {
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(treePathtoFilePath(selPath.toString())));
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        tempSet = (ScenarioElementSet)input.readObject();
                     }
                     catch(ClassNotFoundException classnotfoundException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                     }
                     catch(NullPointerException nullpointerException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read - null pointer", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        input.close();
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     catch(NullPointerException nullpointerException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read - null pointer", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
```

```
                    {
                       //asset
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset"))
                    {

workareaTabbedPane.addTab("Asset:"+tempSet.getelementSetName(),(Asset)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       Asset tempAsset = new Asset();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempAsset = (Asset)tempObject;
                          tempAsset.reInitListeners(datapath,startupScenario);
                          tempAsset.reInitLists();
                          tempAsset.reInitButtons();
                       }
                    }
                    //asset goal
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal"))
                    {

workareaTabbedPane.addTab("Goal:"+tempSet.getelementSetName(),(AssetGoal)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       AssetGoal tempAssetGoal = new AssetGoal();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempAssetGoal = (AssetGoal)tempObject;
                          tempAssetGoal.reInitListeners(datapath,startupScenario);
                          tempAssetGoal.reInitLists();
                          tempAssetGoal.reInitButtons();
                       }
                    }
                    //catalog component
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog Component"))
                    {
                       workareaTabbedPane.addTab("Catalog
Component:"+tempSet.getelementSetName(),(CatalogComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       CatalogComponent tempCatalogComponent = new CatalogComponent();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempCatalogComponent = (CatalogComponent)tempObject;

tempCatalogComponent.reInitListeners(fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileConte
nts_passlength,fileContents_software);
                          tempCatalogComponent.reInitLists(fileContents_software);
                       }
                    }
                    //condition
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition"))
```

```
                    {

workareaTabbedPane.addTab("Condition:"+tempSet.getelementSetName(),(Condition)tempSet.elementContainer.elem
entAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Condition tempCondition = new Condition();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempCondition = (Condition)tempObject;
                           tempCondition.reInitListeners(fileContents_condition);
                        }
                    }
                    //dac group
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group"))
                    {
                        workareaTabbedPane.addTab("DAC
Group:"+tempSet.getelementSetName(),(DACGroup)tempSet.elementContainer.elementAt(tempSet.getlastElementUs
ed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //department
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department"))
                    {

workareaTabbedPane.addTab("Department:"+tempSet.getelementSetName(),(Department)tempSet.elementContainer.e
lementAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //integrity
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity"))
                    {

workareaTabbedPane.addTab("Integrity:"+tempSet.getelementSetName(),(Integrity)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //network
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network"))
                    {

workareaTabbedPane.addTab("Network:"+tempSet.getelementSetName(),(Network)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //physical component
                    if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component"))
                    {
                        workareaTabbedPane.addTab("Physical
Component:"+tempSet.getelementSetName(),(PhysicalComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        PhysicalComponent tempPhys = new PhysicalComponent();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
```

```
                        tempObject = tempSet.elementContainer.get(i);
                        tempPhys = (PhysicalComponent)tempObject;

tempPhys.reInitListeners(datapath,startupScenario,fileContents_basecomp,fileContents_os,fileContents_p asscomplexit
y,fileContents_passlength,fileContents_software);
                        tempPhys.reInitLists(fileContents_software);
                        tempPhys.reInitButtons();
                     }
                  }
                  //secrecy
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy"))
                  {

workareaTabbedPane.addTab("Secrecy:"+tempSet.getelementSetName(),(Secrecy)tempSet.elementContainer.element
At(tempSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                  }
                  //trigger
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger"))
                  {

workareaTabbedPane.addTab("Trigger:"+tempSet.getelementSetName(),(Trigger)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     Trigger tempTrigger = new Trigger();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempTrigger = (Trigger)tempObject;
                        tempTrigger.reInitListeners(datapath,startupScenario,fileContents_trigger);
                     }
                  }
                  //user
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User"))
                  {

workareaTabbedPane.addTab("User:"+tempSet.getelementSetName(),(User)tempSet.elementContainer.elementAt(tem
pSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     User tempUser = new User();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempUser = (User)tempObject;
                        tempUser.reInitListeners(datapath,startupScenario);
                        tempUser.reInitLists();
                        tempUser.reInitButtons();
                     }
                  }
                  //workspace
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace"))
                  {

workareaTabbedPane.addTab("Workspace:"+tempSet.getelementSetName(),(Workspace)tempSet.elementContainer.el
ementAt(tempSet.getlastElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
```

```
                        Workspace tempWorkspace = new Workspace();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempWorkspace = (Workspace)tempObject;
                           tempWorkspace.reInitListeners();
                           tempWorkspace.reInitLists();
                        }
                     }
                     //filter
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter"))
                     {

workareaTabbedPane.addTab("Filter:"+tempSet.getelementSetName(),(Filter)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Filter tempFilter = new Filter();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempFilter = (Filter)tempObject;

tempFilter.reInitListeners(datapath,startupScenario,fileContents_filterblocking,fileContents_filterapps);
                           tempFilter.reInitLists(fileContents_filterapps);
                           tempFilter.reInitButtons();
                        }
                     }
                     //procedural settings
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings"))
                     {
                        workareaTabbedPane.addTab("Procedural
Settings:"+tempSet.getelementSetName(),(ProceduralSettings)tempSet.elementContainer.elementAt(tempSet.getlastEl
ementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempProceduralSettings = (ProceduralSettings)tempObject;
                           tempProceduralSettings.reInitListeners(datapath,startupScenario,
fileContents_passcomplexity, fileContents_passlength, fileContents_passchangefreq);
                           tempProceduralSettings.reInitLists();
                           tempProceduralSettings.reInitButtons();
                        }
                     }
                     //zone
                     if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone"))
                     {

workareaTabbedPane.addTab("Zone:"+tempSet.getelementSetName(),(Zone)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Zone tempZone = new Zone();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
```

```
                        tempZone = (Zone)tempObject;
                        tempZone.reInitListeners(datapath,startupScenario);
                        tempZone.reInitLists();
                        tempZone.reInitButtons();
                     }
                  }
                  //network connection
                  if(treePathtoFilePath(selPath.toString()).startsWith(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection"))
                  {
                     workareaTabbedPane.addTab("Component Network
Connection:"+tempSet.getelementSetName(),(NetworkConnection)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                     tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                     NetworkConnection tempNetworkConnection = new NetworkConnection();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempNetworkConnection = (NetworkConnection)tempObject;
                        tempNetworkConnection.reInitListeners(datapath,startupScenario);
                        tempNetworkConnection.reInitLists();
                        tempNetworkConnection.reInitButtons();
                     }
                  }
                  }
                  catch(ArrayIndexOutOfBoundsException arrayboundException)
                  {
                     JOptionPane.showMessageDialog(this, "Exception during file read - array bound error",
"Exception", JOptionPane.ERROR_MESSAGE);
                  }
               }//right click
               else if(evt.getButton() == evt.BUTTON3 && evt.getClickCount() == 1)
               {
                  rightClickTreePath = selPath.toString();
                  libraryTreePopupMenu.show(evt.getComponent(), evt.getX(), evt.getY());
               }
            }
         }
         //openMenuItemActionPerformed()
         //- creates a file chooser and initializes it to the reuable sets library
         //- tests to see if the file to open is a scenaior (based on file extension)
         //if it is then it opens the file based on the canonical path provided by
          //the file chooser
         //-if the file to be opened is not a scenario (its a descriptor or
         //something that the application should not be opening)
         //-prefetches the object based on the file name provided in the canonical
         //path from the file chooser
         //-based on the descriptor's file extension
         //- create a new tab with the appropriate descriptor title
         //- add the object from the file to the tab manager at a position
         //equal to the tab count -1 (tab count starts at 1 and the tab manager
         //starts at 0 so we always have to subtract 1).  By adding the set
         //to the tab manager at a position equal to the tab count - 1 the
         //two datastructures are always working in parallel to eachother.
         //- get the last element used from the from the set and reinitialize
         //its listeners
         private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt)
         {
            JFileChooser chooser = new JFileChooser();
            chooser.setCurrentDirectory(new File(datapath + "CyberCIEGE/SDT/Reusable Sets Library"));
            Scenario tempScenario = new Scenario(false);
```

```
                ScenarioElementSet tempSet = new ScenarioElementSet();
                Object tempObject = new Object();
                int returnVal = chooser.showOpenDialog(this);
                if(returnVal == JFileChooser.APPROVE_OPTION)
                {
                   //scenario
                   if(chooser.getSelectedFile().getName().endsWith(".CSM"))
                   {
                      try
                      {
                         feedbackTextArea.append("Attempting to open file...\n");
                         feedbackTextArea.append("The File: "+ chooser.getSelectedFile().getCanonicalPath() + " is the
target for opening\n");
                         input = new java.io.ObjectInputStream(new
java.io.FileInputStream(chooser.getSelectedFile().getCanonicalPath()));
                         feedbackTextArea.append("Open complete\n");
                      }
                      catch(IOException ioException)
                      {
                         JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                      }
                      try
                      {
                         feedbackTextArea.append("Attempting to read file...\n");
                         tempScenario = (Scenario)input.readObject();
                         feedbackTextArea.append("Read complete.\n");
                      }
                      catch(ClassNotFoundException classnotfoundException)
                      {
                         JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                      }
                      catch(IOException ioException)
                      {
                         JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                      }
                      try
                      {
                         feedbackTextArea.append("Attempting to close file...\n");
                         input.close();
                         feedbackTextArea.append("Close complete\n");
                      }
                      catch(IOException ioException)
                      {
                         JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                      }
                      workareaTabbedPane.setComponentAt(0, tempScenario);
                      workareaTabbedPane.setTitleAt(0, "Scenario:"+
                               tempScenario.getScenarioFileName());
                      tabManager.setElementAt(tempScenario, 0);
                      startupScenario = tempScenario;
                      startupScenario.reInitListeners();
                      initScenarioTree();
                   }
                   else
                   {
                      try
                      {
                         feedbackTextArea.append("Attempting to open file...\n");
```

```
                        feedbackTextArea.append("The File: "+ chooser.getSelectedFile().getCanonicalPath() + " is the
target for opening\n");
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(chooser.getSelectedFile().getCanonicalPath()));
                        feedbackTextArea.append("Open complete\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to read file...\n");
                        tempSet = (ScenarioElementSet)input.readObject();
                        feedbackTextArea.append("Read complete.\n");
                     }
                     catch(ClassNotFoundException classnotfoundException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to close file...\n");
                        input.close();
                        feedbackTextArea.append("Close complete\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     //asset
                     if(chooser.getSelectedFile().getName().endsWith(".AMS"))
                     {

workareaTabbedPane.addTab("Asset:"+tempSet.getelementSetName(),(Asset)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Asset tempAsset = new Asset();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempAsset = (Asset)tempObject;
                           tempAsset.reInitListeners(datapath,startupScenario);
                           tempAsset.reInitLists();
                           tempAsset.reInitButtons();
                        }
                     }
                     //asset goal
                     if(chooser.getSelectedFile().getName().endsWith(".GMS"))
                     {

workareaTabbedPane.addTab("Goal:"+tempSet.getelementSetName(),(AssetGoal)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
```

```
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        AssetGoal tempAssetGoal = new AssetGoal();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempAssetGoal = (AssetGoal)tempObject;
                           tempAssetGoal.reInitListeners(datapath,startupScenario);
                           tempAssetGoal.reInitLists();
                           tempAssetGoal.reInitButtons();
                        }
                     }
                     //catalog component
                     if(chooser.getSelectedFile().getName().endsWith(".CMS"))
                     {
                        workareaTabbedPane.addTab("Catalog
Component:"+tempSet.getelementSetName(),(CatalogComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        CatalogComponent tempCatalogComponent = new CatalogComponent();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempCatalogComponent = (CatalogComponent)tempObject;

tempCatalogComponent.reInitListeners(fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileConte
nts_passlength,fileContents_software);
                           tempCatalogComponent.reInitLists(fileContents_software);
                        }
                     }
                     //condition
                     if(chooser.getSelectedFile().getName().endsWith(".BMS"))
                     {

workareaTabbedPane.addTab("Condition:"+tempSet.getelementSetName(),(Condition)tempSet.elementContainer.elem
entAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Condition tempCondition = new Condition();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempCondition = (Condition)tempObject;
                           tempCondition.reInitListeners(fileContents_condition);
                        }
                     }
                     //dac group
                     if(chooser.getSelectedFile().getName().endsWith(".DMS"))
                     {
                        workareaTabbedPane.addTab("DAC
Group:"+tempSet.getelementSetName(),(DACGroup)tempSet.elementContainer.elementAt(tempSet.getlastElementUs
ed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     }
                     //department
                     if(chooser.getSelectedFile().getName().endsWith(".EMS"))
                     {

workareaTabbedPane.addTab("Department:"+tempSet.getelementSetName(),(Department)tempSet.elementContainer.e
lementAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
```

```
                    }
                    //filter
                    if(chooser.getSelectedFile().getName().endsWith(".FMS"))
                    {

workareaTabbedPane.addTab("Filter:"+tempSet.getelementSetName(),(Filter)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       Filter tempFilter = new Filter();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempFilter = (Filter)tempObject;

tempFilter.reInitListeners(datapath,startupScenario,fileContents_filterblocking,fileContents_filterapps);
                          tempFilter.reInitLists(fileContents_filterapps);
                          tempFilter.reInitButtons();
                       }
                    }
                    //integrity
                    if(chooser.getSelectedFile().getName().endsWith(".IMS"))
                    {

workareaTabbedPane.addTab("Integrity:"+tempSet.getelementSetName(),(Integrity)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //network
                    if(chooser.getSelectedFile().getName().endsWith(".NMS"))
                    {

workareaTabbedPane.addTab("Network:"+tempSet.getelementSetName(),(Network)tempSet.elementContainer.elemen
tAt(tempSet.getlastElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                    }
                    //physical component
                    if(chooser.getSelectedFile().getName().endsWith(".PM S"))
                    {
                       workareaTabbedPane.addTab("Physical
Component:"+tempSet.getelementSetName(),(PhysicalComponent)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                       tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                       PhysicalComponent tempPhys = new PhysicalComponent();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempPhys = (PhysicalComponent)tempObject;
                          tempPhys.reInitListeners(datapath,startupScenario,
fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileContents_passlength,fileContents_software);
                          tempPhys.reInitLists(fileContents_software);
                          tempPhys.reInitButtons();
                       }
                    }
                    //procedural settings
                    if(chooser.getSelectedFile().getName().endsWith(".LMS"))
                    {
                       workareaTabbedPane.addTab("Procedural
Settings:"+tempSet.getelementSetName(),(ProceduralSettings)tempSet.elementContainer.elementAt(tempSet.getlastEl
ementUsed()));
                       tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
```

```
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = t empSet.elementContainer.get(i);
                           tempProceduralSettings = (ProceduralSettings)tempObject;
                           tempProceduralSettings.reInitListeners(datapath,startupScenario,
fileContents_passcomplexity, fileContents_passlength, fileContents_passchangefreq);
                           tempProceduralSettings.reInitLists();
                           tempProceduralSettings.reInitButtons();
                        }
                     }
                     //secrecy
                     if(chooser.getSelectedFile().getName().endsWith(".SMS"))
                     {

workareaTabbedPane.addTab("Secrecy:"+tempSet.getelementSetName(),(Secrecy)tempSet.elementContainer.element
At(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                     }
                     //trigger
                     if(chooser.getSelectedFile().getName().endsWith(".TMS"))
                     {

workareaTabbedPane.addTab("Trigger:"+tempSet.getelementSetName(),(Trigger)tempSet.elementContainer.elementA
t(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Trigger tempTrigger = new Trigger();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempTrigger = (Trigger)tempObject;
                           tempTrigger.reInitListeners(datapath,startupScenario,fileContents_trigger);
                        }
                     }
                     //user
                     if(chooser.getSelectedFile().getName().endsWith(".UMS"))
                     {

workareaTabbedPane.addTab("User:"+tempSet.getelementSetName(),(User)tempSet.elementContainer.elementAt(tem
pSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        User tempUser = new User();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempUser = (User)tempObject;
                           tempUser.reInitListeners(datapath,startupScenario);
                           tempUser.reInitLists();
                           tempUser.reInitButtons();
                        }
                     }
                     //workspace
                     if(chooser.getSelectedFile().getName().endsWith(".WMS"))
                     {

workareaTabbedPane.addTab("Workspace:"+tempSet.getelementSetName(),(Workspace)tempSet.elementContainer.el
ementAt(tempSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
```

```
                        Workspace tempWorkspace = new Workspace();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempWorkspace = (Workspace)tempObject;
                           tempWorkspace.reInitListeners();
                           tempWorkspace.reInitLists();
                        }
                     }
                     //zone
                     if(chooser.getSelectedFile().getName().endsWith(".ZMS"))
                     {

workareaTabbedPane.addTab("Zone:"+tempSet.getelementSetName(),(Zone)tempSet.elementContainer.elementAt(te
mpSet.getlastElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        Zone tempZone = new Zone();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempZone = (Zone)tempObject;
                           tempZone.reInitListeners(datapath,startupScenario);
                           tempZone.reInitLists();
                           tempZone.reInitButtons();
                        }
                     }
                     //network connection
                     if(chooser.getSelectedFile().getName().endsWith(".KMS"))
                     {
                        workareaTabbedPane.addTab("Component Network
Connection:"+tempSet.getelementSetName(),(NetworkConnection)tempSet.elementContainer.elementAt(tempSet.getla
stElementUsed()));
                        tabManager.insertElementAt(tempSet, (workareaTabbedPane.getTabCount()-1));
                        tempObject = tempSet.elementContainer.elementAt(tempSet.getlastElementUsed());
                        NetworkConnection tempNetworkConnection = new NetworkConnection();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempNetworkConnection = (NetworkConnection)tempObject;
                           tempNetworkConnection.reInitListeners(datapath,startupScenario);
                           tempNetworkConnection.reInitLists();
                           tempNetworkConnection.reInitButtons();
                        }
                     }
                  }
               }

            }
            //setelementdeleteButtonActionPerformed()
            //removes a set element from a reusable set
            //NOTE: the scenario and the 0th element of all sets are ignored by
            //this method.  A set of lenght zero does not make sense for the
            //application and scenarios do not come in sets.
            //-based on the descriptor type as determined by the tab title
            //-if element zero has not been targeted for removal
            //-remove the element from the scenario element sets element container
            //NOTE:the variable deleteElementTarget is used to determine which element
            //to remove.  The value for this variable is initialized in the constructor
            //for this (ScenarioDefiniontool) class and is set in
            //setelementmanagementComboBoxActionPerformed().  After a delete it is
            //set to the previous element in the set
```

```
            //-after the element is removed from the set the tabbed work area is updated
            //and repainted
            private void setelementdeleteButtonActionPerformed(java.awt.event.ActionEvent evt)
            {
               Object tempObject = new Object();
               ScenarioElementSet tempSet = new ScenarioElementSet();
               if(!workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
               {
                  tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                  tempSet = (ScenarioElementSet)tempObject;
                  if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
                  {
                     Asset tempAsset = (Asset)tempSet.elementContainer.get(deleteElementTarget);
                     if(deleteElementTarget != 0)
                     {
                        if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempAsset.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                        {
                           tempSet.elementContainer.remove(deleteElementTarget);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Asset)tempSet.elementContainer.get((deleteElementTarget-1)));
                           deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                           deleteElementTarget = (deleteElementTarget-1);
                           workareaTabbedPaneStateChanged(null);
                           workareaTabbedPane.repaint();
                        }
                     }
                  }
                  if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                  {
                     AssetGoal tempAssetGoal = (AssetGoal)tempSet.elementContainer.get(deleteElementTarget);
                     if(deleteElementTarget != 0)
                     {
                        if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempAssetGoal.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                        {
                           tempSet.elementContainer.remove(deleteElementTarget);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(AssetGoal)tempSet.elementContainer.get((deleteElementTarget-1)));
                           deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                           deleteElementTarget = (deleteElementTarget-1);
                           workareaTabbedPaneStateChanged(null);
                           workareaTabbedPane.repaint();
                        }
                     }
                  }
                  if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                  {
                     CatalogComponent tempCatComp =
(CatalogComponent)tempSet.elementContainer.get(deleteElementTarget);
                     if(deleteElementTarget != 0)
                     {
                        if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempCatComp.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                        {
                           tempSet.elementContainer.remove(deleteElementTarget);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(CatalogComponent)tempSet.elementContainer.get((deleteElementTarget-1)));
```

```
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
               {
                  Condition tempCondition = (Condition)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempCondition.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Condition)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
               {
                  Department tempDepartment = (Department)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempDepartment.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?",
"Delete", JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Department)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }
               if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
               {
                  DACGroup tempDACGroup = (DACGroup)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempDACGroup.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(DACGroup)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
```

```
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }
               if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
               {
                  Filter tempFilter = (Filter)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempFilter.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Filter)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }
               if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
               {
                  Integrity tempIntegrity = (Integrity)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempIntegrity.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Integrity)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }
               if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
               {
                  Network tempNetwork = (Network)tempSet.elementContainer.get(deleteElementTarget);
                  if(deleteElementTarget != 0)
                  {
                     if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempNetwork.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                     {
                        tempSet.elementContainer.remove(deleteElementTarget);
                        workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Network)tempSet.elementContainer.get((deleteElementTarget-1)));
                        deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                        deleteElementTarget = (deleteElementTarget-1);
                        workareaTabbedPaneStateChanged(null);
                        workareaTabbedPane.repaint();
                     }
                  }
               }
```

```
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                    {
                       Secrecy tempSecrecy = (Secrecy)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempSecrecy.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Secrecy)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                    {
                       PhysicalComponent tempPhysComp =
(PhysicalComponent)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempPhysComp.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(PhysicalComponent)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural
Settings"))
                    {
                       ProceduralSettings tempProceduralSettings =
(ProceduralSettings)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempProceduralSettings.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?",
"Delete", JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(ProceduralSettings)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
```

```
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component
Network Connection"))
                    {
                       NetworkConnection tempNetworkConnection =
(NetworkConnection)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempNetworkConnection.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?",
"Delete", JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(NetworkConnection)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                    {
                       Trigger tempTrigger = (Trigger)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempTrigger.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) ==  JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Trigger)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
                    {
                       User tempUser = (User)tempSet.elementContainer.get(deleteElementTarget);
                       if(deleteElementTarget != 0)
                       {
                          if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempUser.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                          {
                             tempSet.elementContainer.remove(deleteElementTarget);
                             workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(User)tempSet.elementContainer.get((deleteElementTarget-1)));
                             deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                             deleteElementTarget = (deleteElementTarget-1);
                             workareaTabbedPaneStateChanged(null);
                             workareaTabbedPane.repaint();
                          }
                       }
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.
                       getSelectedIndex()).startsWith("Workspace"))
                    {
```

```
                    Workspace tempWorkspace = (Workspace)tempSet.elementContainer.
                       get(deleteElementTarget);
                    if(deleteElementTarget != 0)
                    {
                       if(JOptionPane.showConfirmDialog(this,
                          "You are about to delete: " + tempWorkspace.getNameTextField() + " from the " +
tempSet.getelementSetName() + " set.  Are you sure?", "Delete", JOptionPane.INFORMATION_MESSAGE) ==
JOptionPane.YES_OPTION)
                       {
                          tempSet.elementContainer.remove(deleteElementTarget);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Workspace)tempSet.elementContainer.get((deleteElementTarget-1)));
                          deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                          deleteElementTarget = (deleteElementTarget-1);
                          workareaTabbedPaneStateChanged(null);
                          workareaTabbedPane.repaint();

                       }
                    }
                 }
                 if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                 {
                    CatalogComponent tempCatComp =
(CatalogComponent)tempSet.elementContainer.get(deleteElementTarget);
                    if(deleteElementTarget != 0)
                    {
                       if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempCatComp.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                       {
                          tempSet.elementContainer.remove(deleteElementTarget);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(CatalogComponent)tempSet.elementContainer.get((deleteElementTarget-1)));
                          deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                          deleteElementTarget = (deleteElementTarget-1);
                          workareaTabbedPaneStateChanged(null);
                          workareaTabbedPane.repaint();
                       }
                    }
                 }
                 if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                 {
                    Zone tempZone = (Zone)tempSet.elementContainer.get(deleteElementTarget);
                    if(deleteElementTarget != 0)
                    {
                       if(JOptionPane.showConfirmDialog(this, "You are about to delete: " +
tempZone.getNameTextField() + " from the " + tempSet.getelementSetName() + " set.  Are you sure?", "Delete",
JOptionPane.INFORMATION_MESSAGE) == JOptionPane.YES_OPTION)
                       {
                          tempSet.elementContainer.remove(deleteElementTarget);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Zone)tempSet.elementContainer.get((deleteElementTarget-1)));
                          deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                          deleteElementTarget = (deleteElementTarget-1);
                          workareaTabbedPaneStateChanged(null);
                          workareaTabbedPane.repaint();
                       }
                    }
                 }
              }
           }
```

```
        //secrecyMenuItemActionPerformed()
            //- creates a new secrecy object
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a secrecy .SMS to the file
            //name
            //- adds the secrect object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void secrecyMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               Secrecy newSecrecy = new Secrecy();
               newSecrecy.populateStaticSourceLists(fileContents_backgroundcheck);
               ScenarioElementSet tempsecrecyset = new ScenarioElementSet();
               tempsecrecyset.setelementSetName(JOptionPane.showInputDialog("Enter the secrecy file name."));

               if(tempsecrecyset.getelementSetName() != null && tempsecrecyset.getelementSetName() != " " &&
tempsecrecyset.getelementSetName().length() != 0)
               {
                  tempsecrecyset.setelementSetName(tempsecrecyset.getelementSetName()+".SMS");
                  tempsecrecyset.setelementType("Secrecy");
                  tempsecrecyset.elementContainer.add(newSecrecy);

workareaTabbedPane.addTab("Secrecy:"+tempsecrecyset.getelementSetName(),(Secrecy)tempsecrecyset.elementCont
ainer.lastElement());
                  tabManager.insertElementAt(tempsecrecyset, (workareaTabbedPane.getTabCount()-1));
               }
               else
               {
                  JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                  secrecyMenuItemActionPerformed(null);
               }
            }
        //integrityMenuItemActionPerformed()
            //- creates a new integrity object
            //- creates a new scenario element set
            //- prompts for a name for the new set
            //- tests the name provided to make sure it is legal
            //- appends the file extension for a integrity .IMS to the file
            //name
            //- adds the integrity object to the scenario element set
            //- creates a new tab in the tabbed work area, names it and adds the
            //new object to it
            //- if the file name provided was not legal the method prompts for
            //input until it is legal
            private void integrityMenuItemActionPerformed(java.awt.event.ActionEvent evt)
            {
               Integrity newIntegrity = new Integrity();
               newIntegrity.populateStaticSourceLists(fileContents_backgroundcheck);
               ScenarioElementSet tempintegrityset = new ScenarioElementSet();
               tempintegrityset.setelementSetName(JOptionPane.showInputDialog("Enter the integrity file name."));

               if(tempintegrityset.getelementSetName() != null && tempintegrityset.getelementSetName() != " " &&
tempintegrityset.getelementSetName().length() != 0)
               {
                  tempintegrityset.setelementSetName(tempintegrityset.getelementSetName()+".IMS");
                  tempintegrityset.setelementType("Integrity");
                  tempintegrityset.elementContainer.add(newIntegrity);
```

```
workareaTabbedPane.addTab("Integrity:"+tempintegrityset.getelementSetName(),(Integrity)tempintegrityset.elementC
ontainer.lastElement());
                    tabManager.insertElementAt(tempintegrityset, (workareaTabbedPane.getTabCount()-1));
                }
                else
                {
                    JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                    integrityMenuItemActionPerformed(null);
                }
            }
        //workareaTabbedPaneStateChanged()
            //this method is responsible for populating the set element management
            //combo box on the tool bar
            //-if the tabbed pane tab is not a scenario tab
            //-get the object associated with the selected tab from the tab manager
            //-prep the combo box
            //-based on the descriptor type as obtained from the tab name
            //-for each element in the set obtained from the tab manager
            //-get the individual form from the scenario element set and add its name
            //to the combo box
            private void workareaTabbedPaneStateChanged(javax.swing.event.ChangeEvent evt)
            {
               Object tempObject = new Object();
               Object tempElementObject = new Object();
               ScenarioElementSet tempSet = new ScenarioElementSet();
               setelementmanagementComboBox.removeAllItems();
               if(!workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
               {
                  tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                  tempSet = (ScenarioElementSet)tempObject;
                  setelementmanagementComboBox.removeAllItems();
                  setelementmanagementComboBox.addItem(makeObj(" "));
                  setelementmanagementComboBox.addItem(makeObj("New"));
                  setelementmanagementComboBox.addItem(makeObj(" "));
                  for(int i = 0; i < tempSet.getSize(); i++)
                  {
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
                     {
                        Asset tempAsset = new Asset();
                        tempElementObject = tempSet.elementContainer.get(i);
                        tempAsset = (Asset)tempElementObject;
                        setelementmanagementComboBox.addItem(makeObj(tempAsset.getNameTextField()));
                     }
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                     {
                        AssetGoal tempAssetGoal = new AssetGoal();
                        tempElementObject = tempSet.elementContainer.get(i);
                        tempAssetGoal = (AssetGoal)tempElementObject;
                        setelementmanagementComboBox.addItem(makeObj(tempAssetGoal.getNameTextField()));
                     }
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                     {
                        CatalogComponent tempCatalog = new CatalogComponent();

tempCatalog.reInitListeners(fileContents_basecomp,fileContents_os,fileContents_passcomplexity,fileContents_passlen
gth,fileContents_software);
                        tempElementObject = tempSet.elementContainer.get(i);
                        tempCatalog = (CatalogComponent)tempElementObject;
                        setelementmanagementComboBox.addItem(makeObj(tempCatalog.getNameTextField()));
```

```
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
                    {
                       Condition tempCondition = new Condition();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempCondition = (Condition)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempCondition.getNameTextField()));
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                    {
                       DACGroup tempDACGroup = new DACGroup();
                       tempDACGroup.reInitListeners(fileContents_backgroundcheck);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempDACGroup = (DACGroup)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempDACGroup.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                    {
                       Department tempDepartment = new Department();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempDepartment = (Department)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempDepartment.getNameTextField()));
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
                    {
                       Filter tempFilter = new Filter();
                       tempFilter.reInitListeners(datapath,startupScenario,
                          fileContents_filterblocking,fileContents_filterapps);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempFilter = (Filter)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempFilter.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
                    {
                       Integrity tempIntegrity = new Integrity();
                       tempIntegrity.reInitListeners(fileContents_backgroundcheck);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempIntegrity = (Integrity)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempIntegrity.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                    {
                       Network tempNetwork = new Network();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempNetwork = (Network)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempNetwork.getNameTextField()));
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                    {
                       PhysicalComponent tempPhysical = new PhysicalComponent();
                       tempPhysical.reInitListeners(datapath, startupScenario, fileContents_basecomp,
                       fileContents_os,fileContents_passcomplexity,fileContents_passlengt h,
                       fileContents_software);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempPhysical = (PhysicalComponent)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempPhysical.getNameTextField()));
```

```
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural Settings"))
                    {
                       ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                       tempProceduralSettings.reInitListeners(datapath,startupScenario,
                          fileContents_passcomplexity, fileContents_passlength,
                          fileContents_passchangefreq);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempProceduralSettings = (ProceduralSettings)tempElementObject;

setelementmanagementComboBox.addItem(makeObj(tempProceduralSettings.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component Network
Connection")) {
                       NetworkConnection tempNetworkConnection = new NetworkConnection();
                       tempNetworkConnection.reInitListeners(datapath,startupScenario);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempNetworkConnection = (NetworkConnection)tempElementObject;

setelementmanagementComboBox.addItem(makeObj(tempNetworkConnection.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                    {
                       Secrecy tempSecrecy = new Secrecy();
                       tempSecrecy.reInitListeners(fileContents_backgroundcheck);
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempSecrecy = (Secrecy)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempSecrecy.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger")) {
                       Trigger tempWorkspace = new Trigger();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempWorkspace = (Trigger)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempWorkspace.getNameTextField()));
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User")) {
                       User tempUser = new User();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempUser = (User)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempUser.getNameTextField()));
                    }

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Workspace"))
                    {
                       Workspace tempWorkspace = new Workspace();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempWorkspace = (Workspace)tempElementObject;
                       tempWorkspace.getNameTextField();
                       setelementmanagementComboBox.addItem(makeObj(tempWorkspace.getNameTextField()));
                    }
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                    {
                       Zone tempZone = new Zone();
                       tempElementObject = tempSet.elementContainer.get(i);
                       tempZone = (Zone)tempElementObject;
                       setelementmanagementComboBox.addItem(makeObj(tempZone.getNameTextField()));
                    }
                 }
```

```
        }
    }
//setelementmanagementComboBoxActionPerformed()
    //this method provides the functionality to the set element
    //management combo box.  The 2 purposes of this combo box is to
    //1).  Add new elements to a set
    //2).  change to a different element in the set
    //the method is laid out with purpose (1) first and (2) second.
    //TO ADD A NEW ELEMENT
    //-if new is selected based on the type of descriptor as determined by
    //the selected tabs name
    //-get the object that corresponds to the selected tab
    //-cast it to a scenario element set
    //-get the last element in the set
    //-check its name
    //-if it has been named
    //-create a new descriptor of the type specifed before
    //-initialize its listeners (as necessary)
    //-add it to the set
    //-set the tabbed work area to the new object.
    //TO CHANGE ELEMENTS
    //-if the choice made in the combo box is not one of the blank entries or
    //new
    //-if the tab selected is not the scenairo tab
    //-based on the descriptor type as determined by the selcted tabs name
    //-set the form from the scenairo element set based on the results of
    //the ternary operator.
    //the ternary operator basically says:
        //if the combo box selected index is greater than 3 (the 3 is to bypass
        //the blank the new and the other blank in the combo box)
        //return the selected index - 3 (three is used again for the reason
        //mentioned above) or 0
    //-set the delete tab and delete element values
    //-repaint the gui
    private void setelementmanagementComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        Asset tempAsset = new Asset();
        AssetGoal tempAssetGoal = new AssetGoal();
        CatalogComponent tempCatalog = new CatalogComponent();
        DACGroup tempDACGroup = new DACGroup();
        Department tempDepartment = new Department();
        Condition tempCondition = new Condition();
        Filter tempFilter = new Filter();
        Integrity tempIntegrity = new Integrity();
        Network tempNetwork = new Network();
        PhysicalComponent tempPhysical = new PhysicalComponent();
        ProceduralSettings tempProceduralSettings = new ProceduralSettings();
        NetworkConnection tempNetworkConnection = new NetworkConnection();
        Secrecy tempSecrecy = new Secrecy();
        Trigger tempTrigger = new Trigger();
        User tempUser = new User();
        Workspace tempWorkspace = new Workspace();
        Zone tempZone = new Zone();
        Object tempObject = new Object();
        ScenarioElementSet tempSet = new ScenarioElementSet();
        if(String.valueOf(setelementmanagementComboBox.getSelectedItem()) == "New")
        {
            //asset
            if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
            {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
```

```
                        tempObject = tempSet.elementContainer.lastElement();
                        tempAsset = (Asset)tempObject;
                        if(!(tempAsset.getNameTextField().length() == 0))
                        {
                           tempAsset = new Asset();
                           tempAsset.reInitListeners(datapath,startupScenario);
                           tempSet.elementContainer.addElement(tempAsset);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Asset)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //asset goal
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempObject = tempSet.elementContainer.lastElement();
                        tempAssetGoal = (AssetGoal)tempObject;
                        if(!(tempAssetGoal.getNameTextField().length() == 0))
                        {
                           tempAssetGoal = new AssetGoal();
                           tempAssetGoal.reInitListeners(datapath,startupScenario);
                           tempSet.elementContainer.addElement(tempAssetGoal);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(AssetGoal)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //catalog component
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempObject = tempSet.elementContainer.lastElement();
                        tempCatalog = (CatalogComponent)tempObject;
                        if(!(tempCatalog.getNameTextField().length() == 0))
                        {
                           tempCatalog = new CatalogComponent();
                           tempCatalog.reInitListeners(
                                 fileContents_basecomp,fileContents_os,
                                 fileContents_passcomplexity,
                                 fileContents_passlength,fileContents_software);
                           tempSet.elementContainer.addElement(tempCatalog);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(CatalogComponent)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
```

```
                    //condition

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
                    {
                       tempObject = t abManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                       tempCondition = (Condition)tempObject;
                       if(!(tempCondition.getNameTextField().length() == 0))
                       {
                          tempCondition = new Condition();
                          tempCondition.reInitListeners(fileContents_condition);
                          tempSet.elementContainer.addElement(temp Condition);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Condition)tempSet.elementContainer.lastElement());
                       }
                       else
                       {
                          JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                       }
                    }
                    //dac group
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                       tempDACGroup = (DACGroup)tempObject;
                       if(!(tempDACGroup.getNameTextField().length() == 0))
                       {
                          tempDACGroup = new DACGroup();
                          tempDACGroup.reInitListeners(fileContents_backgroundcheck);
                          tempSet.elementContainer.addElement(tempDACGroup);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(DACGroup)tempSet.elementContainer.lastElement());
                       }
                       else
                       {
                          JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                       }
                    }
                    //department

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.element Container.lastElement();
                       tempDepartment = (Department)tempObject;
                       if(!(tempDepartment.getNameTextField().length() == 0))
                       {
                          tempDepartment = new Department();
                          tempSet.elementContainer.addElement(tempDepartment);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Department)tempSet.elementContainer.lastElement());
                       }
                       else
                       {
```

```
                        JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                //filter
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempObject = tempSet.elementContainer.lastElement();
                   tempFilter = (Filter)tempObject;
                   if(!(tempFilter.getNameTextField().length() == 0))
                   {
                      tempFilter = new Filter();
                      tempFilter.reInitListeners(datapath,startupScenario,
                         fileContents_filterblocking,fileContents_filterapps);
                      tempSet.elementContainer.addElement(tempFilter);
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Filter)tempSet.elementContainer.lastElement());
                   }
                   else
                   {
                      JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                   }
                }
                //integrity
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempObject = tempSet.elementContainer.lastElement();
                   tempIntegrity = (Integrity)tempObject;
                   if(!(tempIntegrity.getNameTextField().length() == 0))
                   {
                      tempIntegrity = new Integrity();
                      tempIntegrity.reInitListeners(fileContents_backgroundcheck);
                      tempSet.elementContainer.addElement(tempIntegrity);
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Integrity)tempSet.elementContainer.lastElement());
                   }
                   else
                   {
                      JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                   }
                }
                //network
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   tempObject = tempSet.elementContainer.lastElement();
                   tempNetwork = (Network)tempObject;
                   if(!(tempNetwork.getNameTextField().length() == 0))
                   {
                      tempNetwork = new Network();
                      tempSet.elementContainer.addElement(tempNetwork);
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Network)tempSet.elementContainer.lastElement());
                   }
                   else
```

```
                        {
                            JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                    }
                    //physical component
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                       tempPhysical = (PhysicalComponent)tempObject;
                       if(!(tempPhysical.getNameTextField().length() == 0))
                       {
                          tempPhysical = new PhysicalComponent();
                          tempPhysical.reInitListeners(datapath,startupScenario,
                                  fileContents_basecomp,fileContents_os,
                                  fileContents_passcomplexity,
                                  fileContents_passlength,fileContents_software);
                          tempSet.elementContainer.addElement(tempPhysical);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(PhysicalComponent)tempSet.elementContainer.lastElement());
                       }
                       else
                       {
                          JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                       }
                    }
                    //procedural settings
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural
Settings"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                       tempProceduralSettings = (ProceduralSettings)tempObject;
                       if(!(tempProceduralSettings.getNameTextField().length() == 0))
                       {
                          tempProceduralSettings = new ProceduralSettings();
                          tempProceduralSettings.reInitListeners(datapath,startupScenario,
                             fileContents_passcomplexity, fileContents_passlength,
                             fileContents_passchangefreq);
                          tempSet.elementContainer.addElement(tempProceduralSettings);
                          workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(ProceduralSettings)tempSet.elementContainer.lastElement());
                       }
                       else
                       {
                          JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                       }
                    }
                    //network connection
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component
Network Connection"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                       tempNetworkConnection = (Net workConnection)tempObject;
```

```
                        if(!(tempNetworkConnection.getNameTextField().length() == 0))
                        {
                           tempNetworkConnection = new NetworkConnection();
                           tempNetworkConnection.reInitListeners(datapath,startupScenario);
                           tempSet.elementContainer.addElement(tempNetworkConnection);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(NetworkConnection)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //secrecy
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempObject = tempSet.elementContainer.lastElement();
                        tempSecrecy = (Secrecy)tempObject;
                        if(!(tempSecrecy.getNameTextField().length() == 0))
                        {
                           tempSecrecy = new Secrecy();
                           tempSecrecy.reInitListeners(fileContents_backgroundcheck);
                           tempSet.elementContainer.addElement(tempSecrecy);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Secrecy)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //trigger
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempObject = tempSet.elementContainer.lastElement();
                        tempTrigger = (Trigger)tempObject;
                        if(!(tempTrigger.getNameTextField().length() == 0))
                        {
                           tempTrigger = new Trigger();
                           tempTrigger.reInitListeners(datapath,startupScenario,
                                                       fileContents_trigger);
                           tempSet.elementContainer.addElement(tempTrigger);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Trigger)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //user
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
```

```
                        tempObject = tempSet.elementContainer.lastElement();
                        tempUser = (User)tempObject;
                        if(!(tempUser.getNameTextField().length() == 0))
                        {
                           tempUser = new User();
                           tempUser.reInitListeners(datapath, startupScenario);
                           tempSet.elementContainer.addElement(tempUser);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(User)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //workspace

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Workspace"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempObject = tempSet.elementContainer.lastElement();
                        tempWorkspace = (Workspace)tempObject;
                        if(!(tempWorkspace.getNameTextField().length() == 0))
                        {
                           tempWorkspace = new Workspace();
                           tempSet.elementContainer.addElement(tempWorkspace);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Workspace)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //zone
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                       tempObject = tempSet.elementContainer.lastElement();
                        tempZone = (Zone)tempObject;
                        if(!(tempZone.getNameTextField().length() == 0))
                        {
                           tempZone = new Zone();
                           tempZone.reInitListeners(datapath, startupScenario);
                           tempSet.elementContainer.addElement(tempZone);
                           workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Zone)tempSet.elementContainer.lastElement());
                        }
                        else
                        {
                           JOptionPane.showMessageDialog(this, "The set element must be named first.", "Name Error",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                  }
                  if(String.valueOf(setelementmanagementComboBox.getSelectedItem()) != "New" &&
String.valueOf(setelementmanagementComboBox.getSelectedItem()) != " " &&
!workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
```

```
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   //asset
                   if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
                   {
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Asset)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                      deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                      deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                      workareaTabbedPane.repaint();
                   }
                   //asset goal
                   if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                   {
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(AssetGoal)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                      deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                      deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                      workareaTabbedPane.repaint();
                   }
                   //catalog component
                   if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                   {
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(CatalogComponent)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                      deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                      deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                      workareaTabbedPane.repaint();
                   }
                   //condition

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
                   {
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Condition)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                      deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                      deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                      workareaTabbedPane.repaint();
                   }
                   //dac group
                   if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                   {
                      workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(DACGroup)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                      deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                      deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                      workareaTabbedPane.repaint();
                   }
                   //department
```

```
if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                    {
                       workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Department)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                       deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                       deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                       workareaTabbedPane.repaint();
                    }
                    //filter
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
                    {
                       workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Filter)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                       deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                       deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                       workareaTabbedPane.repaint();
                    }
                    //integrity
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
                    {
                       workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Integrity)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                       deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                       deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                       workareaTabbedPane.repaint();
                    }
                    //network
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                    {
                       workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Network)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                       deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                       deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                       workareaTabbedPane.repaint();
                    }
                    //physical component
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                    {
                       workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(PhysicalComponent)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                       deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                       deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                       workareaTabbedPane.repaint();
                    }
                    //procedural settings
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural
Settings"))
                    {
```

```
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(ProceduralSettings)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //network connection
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component
Network Connection"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(NetworkConnection)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //secrecy
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Secrecy)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //trigger
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Trigger)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //user
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(User)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //workspace

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Workspace"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Workspace)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
```

```
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
                //zone
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                {
                    workareaTabbedPane.setComponentAt(workareaTabbedPane.getSelectedIndex(),
(Zone)tempSet.elementContainer.get(((setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0))));
                    deleteTabTarget = workareaTabbedPane.getSelectedIndex();
                    deleteElementTarget = (setelementmanagementComboBox.getSelectedIndex() >
3)?(setelementmanagementComboBox.getSelectedIndex()-3):(0);
                    workareaTabbedPane.repaint();
                }
            }
        }
      //libraryTreeTreeExpanded()
        //this method causes the reusable set library tree to be populated
        private void libraryTreeTreeExpanded(javax.swing.event.TreeExpansionEvent evt)
        {
            TreePath path = evt.getPath();
            FileNode node = (FileNode)path.getLastPathComponent();
            if( ! node.isExplored()) {
                DefaultTreeModel model = (DefaultTreeModel)libraryTree.getModel();
//                feedbackTextArea.setText("exploring ...\n");
                node.explore();
                model.nodeStructureChanged(node);
            }
        }
      // buildMenuItemActionPerformed()
        //This method is the method that actually causes the SDF to be written
        //it counts on the toString() method of EVERY descriptor to be working
        //and correct.
        //This method writes a few key elements of the SDF but no where near
        //as much as the individual toString() methods
        //this method is so complex that it is commented throughout
        private void buildMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {
            Object tempObject = new Object();
            String path_to_file = new String();
            path_to_file = datapath + "CyberCIEGE/SDT/Reusable Sets Library/";
            ScenarioElementSet tempSet = new ScenarioElementSet();
            java.util.Vector tempVector = new java.util.Vector();
            Asset tempAsset = new Asset();
            AssetGoal tempAssetGoal = new AssetGoal();
            CatalogComponent tempCatComp = new CatalogComponent();
            Condition tempCondition = new Condition();
            DACGroup tempDACGroup = new DACGroup();
            Department tempDept = new Department();
            Integrity tempInt = new Integrity();
            Network tempNet = new Network();
            PhysicalComponent tempPhysComp = new PhysicalComponent();
            Secrecy tempSec = new Secrecy();
            Trigger tempTrigger = new Trigger();
            User tempUser = new User();
            Workspace tempWorkspace = new Workspace();
            Zone tempZone = new Zone();
            String tempStr = startupScenario.getScenarioFileName();
            tempStr = tempStr.replaceAll(".CSM", " ");
            tempStr = tempStr.trim();
            try
```

```
                {
                   buildOutput = new PrintWriter(new FileOutputStream(datapath +
"CyberCIEGE/SDT/SDF/"+tempStr +".SDF"));
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                buildOutput.print("//FILE:" + startupScenario.getScenarioFileName() + "\n"+ "//DESIGNER:" +
startupScenario.getDesignerName() + "\n");
                //the output below causes organization & site
                buildOutput.print(startupScenario.toString(null));
                //do not delete the buldOrder array - if you do the scenario will not
                //only not build - it will not build right even if it does.  This array
                //reorders the scenario sets so that they build in the order set in the
                //SFT
                //The numbers in the buildOrder array correspond to the
                //index numbers of the scenario manager e.g. Zone is index 13
                //asset is index 0 etc.  See the comments in Scenario for more info.
                //The SFT specfies the following order:
                //Organization ---------------- see Scenario.toString() - not in the buildOrder array
                //Site ------------------------ see Scenario.toString() - not in the buildOrder array
                //Zone -----------------------------see zone.toString() - buildOrder[0]=13
                //Department ---------------- see Department.toString() - buildOrder[1]=5
                //Network ---------------------- see Network.toString() - buildOrder[2]=7
                //Secrecy ---------------------- see Secrecy.toString() - buildOrder[3]=9
                //Integrity ------------------ see Integrity.toString() - buildOrder[4]=6
                //DACGroup -------------------- see DACGroup.toString() - buildOrder[5]=4
                //Asset ------------------------- see Asset.toString() - buildOrder[6]=0
                //AssetGoal ------------------ see AssetGoal.toString() - buildOrder[7]=1
                //User --------------------------- see User.toString() - buildOrder[8]=11
                //Workspace ------------------ has no output in the SDF - buildOrder[9]=12
                //PhysicalComponent - see PhysicalComponent.toString() - buildOrder[10]=8
                //CatalogComponent --- see CatalogComponent.toString() - buildOrder[11]=2
                //Other ------------------------- built by the descriptors that use them
                //Condition ------------------ see Condition.toString() - buildOrder[13]=3
                //Trigger ---------------------- see Trigger.toString() - buildOrder[14]=10
                int buildOrder[] = {13,5,7,9,6,4,0,1,11,12,8,2,14,3,10};
                //the basic action of build is to say
                //for each of the descriptor vectors in scenario manager (for i)
                //then for each of the sets added to that descriptors vector (for j)
                //based on the first switch looking at buildOrder[i]
                //fetch the file named for the scenairo element set
                //then for each member of that set (for k)
                //based on the second switch also looking at buildOrder[i]
                //call the toString() for that object
                for(int i = 0; i < 15; i++)//major set vectors e.g. network, secrecy ...
                {
                   //skip index 12 (vector 14) it is not going to be processed this
                   //way -
                   //14 is the Other vector(filter, procedural settings) and its
                   //contents will be called from inside the toString() that needs
                   //them - index 9 points to vector 12 - workspace this has been
                   //removed from the SFT
                   if(i != 12)
                   {
                      tempObject = startupScenario.scenarioManager.get(buildOrder[i]);
                      tempVector = (java.util.Vector)tempObject;
                   }
                   if(i == 12)
                   {
```

```
                        //the ouput below causes briefing, debriefwin & debrieflose
                        //to be written to the file
                        buildOutput.print(startupScenario.toStringPost(null));
                    }
                    //DACGroups build is a little different than the other descriptors
                    //as a result some of it will be built here and the rest each
                    //DACGroup set.  The following strings are used in case: 4
                    //I acknowledge that this is a cheat but it allows the user to
                    //use the interface the same for DACGroup as everyother
                    //descriptor
                    String dac_pre = new String();
                    String dac_post = new String();
                    dac_pre += "DACGroups:\n\t//A listing of the DAC groups plus"+
                    " initial background check levels.  Background\n\t//check values"+
                    " are: \"none\",\"low\",\"medium\" and \"high\".\n\n";
                    dac_post +=":end     //of DACGroups\n\n";
                    String condition_pre = new String();
                    String condition_post = new String();
                    condition_pre += "//The following are a set of conditions and triggers. A trigger is triggered
when\n//the condition list bound to the trigger evaluates to true.\n\nConditions:\n";
                    condition_post +=":end //Of conditions block\n\n";
                    String trigger_pre = new String();
                    String trigger_post = new String();
                    trigger_pre += "//The trigger block contains all the triggers in the scenario.\nTriggers:\n";
                    trigger_post +=":end //Of triggers block\n\n";
                    for(int j = 0; j < tempVector.size(); j++)
                    {
                       tempObject = tempVector.get(j);//gets the scenario element set/descriptor
                       //file name saved when the set was added to the scenario
                       //do some file IO to get the file referred to by tempObject
                       switch(buildOrder[i])
                       {
                          case 0://Asset
                             try
                             {
                                input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Asset/"+tempObject.toString()));
                             }
                             catch(IOException ioException)
                             {
                                JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                             }
                             try
                             {
                                tempSet = (ScenarioElementSet)input.readObject();
                             }
                             catch(ClassNotFoundException classnotfoundException)
                             {
                                JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                             }
                             catch(IOException ioException)
                             {
                                JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                             }
                             try
                             {
                                input.close();
                             }
                             catch(IOException ioException)
```

```
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                    case 1://AssetGoal
                        try
                        {
                            input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Goal/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            input.close();
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                    case 2://Catalog Component
                        try
                        {
                            input = new java.io.ObjectInputStream(new java.io.FileInputStream(path_to_file+"Catalog
Component/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        input.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                case 3://Condition
                    try
                    {
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Condition/"+tempObject.toString()));
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        input.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                case 4://DAC Group
                    try
                    {
                        input = new java.io.ObjectInputStream(new java.io.FileInputStream(path_to_file+"DAC
Group/"+tempObject.toString()));
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        tempSet = (ScenarioElementSet)input.readObject();
```

```
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                     case 5://Department
                        try
                        {
                           input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Department/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                     case 6://Integrity
                        try
                        {
                           input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Integrity/"+tempObject.toString()));
                        }
```

```
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                     case 7://Network
                        try
                        {
                           input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Network/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
```

```
                        }
                        break;
                    case 8://Physical Component
                        try
                        {
                            input = new java.io.ObjectInputStream(new java.io.FileInputStream(path_to_file+"Physical
Component/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            input.close();
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                    case 9://Secrecy
                        try
                        {
                            input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Secrecy/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                            tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
```

```
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                     case 10://Trigger
                        try
                        {
                           input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Trigger/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           input.close();
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        break;
                     case 11://User
                        try
                        {
                           input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"User/"+tempObject.toString()));
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           tempSet = (ScenarioElementSet)input.readObject();
                        }
                        catch(ClassNotFoundException classnotfoundException)
                        {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        input.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                case 12://Workspace
                    break;
                case 13://Zone
                    try
                    {
                        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(path_to_file+"Zone/"+tempObject.toString()));
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during input stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read - the object was not
found", "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        input.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    break;
                }//end of switch 1
            for(int k = 0; k < tempSet.elementContainer.size(); k++)
            {
                switch(buildOrder[i])
                {
                    case 0://Asset
                        tempObject = tempSet.elementContainer.get(k);
```

```
    tempAsset = (Asset)tempObject;
    buildOutput.print(tempAsset.toString(null));
    break;
case 1://AssetGoal
    tempObject = tempSet.elementContainer.get(k);
    tempAssetGoal = (AssetGoal)tempObject;
    buildOutput.print(tempAssetGoal.toString(null));
    break;
case 2://Catalog Component
    tempObject = tempSet.elementContainer.get(k);
    tempCatComp = (CatalogComponent)tempObject;
    buildOutput.print(tempCatComp.toString(null));
    break;
case 3://Condition
    tempObject = tempSet.elementContainer.get(k);
    tempCondition = (Condition)tempObject;
    buildOutput.print(condition_pre);
    buildOutput.print(tempCondition.toString(null));
    buildOutput.print(condition_post);
    break;
case 4://DAC Group
    tempObject = tempSet.elementContainer.get(k);
    tempDACGroup = (DACGroup)tempObject;
    buildOutput.print(dac_pre);
    buildOutput.print(tempDACGroup.toString(null));
    buildOutput.print(dac_post);
    break;
case 5://Department
    tempObject = tempSet.elementContainer.get(k);
    tempDept = (Department)tempObject;
    buildOutput.print(tempDept.toString(null));
    break;
case 6://Integrity
    tempObject = tempSet.elementContainer.get(k);
    tempInt = (Integrity)tempObject;
    buildOutput.print(tempInt.toString(null));
    break;
case 7://Network
    tempObject = tempSet.elementContainer.get(k);
    tempNet = (Network)tempObject;
    buildOutput.print(tempNet.toString(null));
    break;
case 8://Physical Component
    //NOTE!!! when you do a physical component build
    //you have to pass the datapath so that file IO can
    //be perfomed for the procedural settings &
    //network connections toString inside of
    //physical component
    tempObject = tempSet.elementContainer.get(k);
    tempPhysComp = (PhysicalComponent)tempObject;
    buildOutput.print(tempPhysComp.toString(datapath));
    break;
case 9://Secrecy
    tempObject = tempSet.elementContainer.get(k);
    tempSec = (Secrecy)tempObject;
    buildOutput.print(tempSec.toString(null));
    break;
case 10://Trigger
    tempObject = tempSet.elementContainer.get(k);
    tempTrigger = (Trigger)tempObject;
    buildOutput.print(trigger_pre);
    buildOutput.print(tempTrigger.toString(null));
```

```
                        buildOutput.print(trigger_post);
                        break;
                     case 11://User
                        tempObject = tempSet.elementContainer.get(k);
                        tempUser = (User)tempObject;
                        buildOutput.print(tempUser.toString(null));
                        break;
//                   case 12://Workspace
//                      tempObject = tempSet.elementContainer.get(k);
//                      tempWorkspace = (Workspace)tempObject;
//                      buildOutput.print(tempWorkspace.toString(null));
//                      break;
                     case 13://Zone
                        //NOTE!!! when you do a zone build you have to
                        //pass the datapath so that file IO can
                        //be perfomed for the procedural settings
                        //toString inside of zone
                        tempObject = tempSet.elementContainer.get(k);
                        tempZone = (Zone)tempObject;
                        buildOutput.print(tempZone.toString(datapath));
                        break;
                  }//end of switch 2
               }//end of for(int k = 0...
            }//end of for(int j = 0...
         }//end of for(int i = 0...
         buildOutput.close();
      }
      //exitMenuItemActionPerformed()
      //quits the application and kills ALL threads of execution
      private void exitMenuItemActionPerformed(java.awt.event.ActionEvent evt)
      {
         System.exit(0);
      }
      //savescenarioasMenuItemActionPerformed()
      //this method allows the user to save the current scenario under a
      //different name
      //if the selected tab is the scenario tab it prompts for the new file name
      //then it checks for a legal name
      //then it creates an output object to write the file to
      private void savescenarioasMenuItemActionPerformed(java.awt.event.ActionEvent evt)
      {
         Object tempObject = new Object();
         Scenario tempScenario;
         ScenarioElementSet tempSet = new ScenarioElementSet();
         if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
         {
            String scenarioSaveAsFileName = JOptionPane.showInputDialog("Enter the new scenario file
name.");
            tempObject = tabManager.get(0);
            tempScenario = (Scenario)tempObject;
            tempScenario.setScenarioFileName(scenarioSaveAsFileName+".CSM");
            if(tempScenario.getScenarioFileName() != null && tempScenario.getScenarioFileName() != " " &&
tempScenario.getScenarioFileName().length() != 0)
            {
               feedbackTextArea.append("Request to save the Scenario " + tempScenario.getScenarioFileName()
+ " is proccessing...\n");
               try
               {
                  feedbackTextArea.append("Attempting to open file...\n");
                  output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()));
```

```
                        feedbackTextArea.append("Save target is: " + datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempScenario);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                else
                {
                    JOptionPane.showMessageDialog(this, "Unable to save - a new Scenario File Name is required",
"Scenario File Name Error", JOptionPane.ERROR_MESSAGE);
                    savescenarioasMenuItemActionPerformed(null);
                }
            }
        }
    //saveasMenuItemActionPerformed()
        //this method allows the scenario element set in the currently selected
        //tab to be saved under a new name
        //if the active tab is not the scenario tab
        //prompt for a new name
        //based on the descriptor type as determiend by the name of the selected tab
        //get the object associtaed with the active tab from the tab manager
        //set the sets name to the new name
        //update the feedback area
        //create an output object to write to
        //write the file
        //reinit any lists and listeners that stopped working as a result of the
        //save (do this as needed).
        private void saveasMenuItemActionPerformed(java.awt.event.ActionEvent evt)
        {
            String setSaveAsFileName = "untitled";
            if(!workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
            {
                setSaveAsFileName = JOptionPane.showInputDialog("Enter the new reuseable set file name.");
            }
             if(setSaveAsFileName != null && setSaveAsFileName != " " && setSaveAsFileName.length() != 0)
                {
```

```
                    Object tempObject = new Object();
                    ScenarioElementSet tempSet = new ScenarioElementSet();
                    //asset
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempSet.setelementSetName(setSaveAsFileName+".AMS");
                       feedbackTextArea.append("Request to save the Asset set " + tempSet.getelementSetName() + "
is proccessing...\n");
                       try
                       {
                          feedbackTextArea.append("Attempting to open file...\n");
                          output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Asset/"+tempSet.getelementSetName()));
                          feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset/"+tempSet.getelementSetName()+"\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                       }
                       try
                       {
                          feedbackTextArea.append("Attempting to write file...\n");
                          output.writeObject(tempSet);
                          output.flush();
                          feedbackTextArea.append("File written.\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
                       try
                       {
                          feedbackTextArea.append("Attempting to close file...\n");
                          output.close();
                          feedbackTextArea.append("File closed.\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
                       initLibraryTree();
                       Asset tempAsset = new Asset();
                       for(int i = 0; i < tempSet.elementContainer.size(); i++)
                       {
                          tempObject = tempSet.elementContainer.get(i);
                          tempAsset = (Asset)tempObject;
                          tempAsset.reInitLists();
                          tempAsset.reInitListeners(datapath, startupScenario);
                       }
                    }
                    //asset goal
                    if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempSet.setelementSetName(setSaveAsFileName+".GMS");
```

```
                        feedbackTextArea.append("Request to save the Goal set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Goal/"+tempSet.getelementSetName()));
                           feedbackText Area.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        AssetGoal tempAssetGoal = new AssetGoal();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempAssetGoal = (AssetGoal)tempObject;
                           tempAssetGoal.reInitLists();
                           tempAssetGoal.reInitListeners(datapath, startupScenario);
                        }
                     }
                     //catalog component
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.
                           getSelectedIndex()).startsWith("Catalog Component"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.
                                                getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".CMS");
                        feedbackTextArea.append("Request to save the Catalog Component set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
```

```
                        FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets Library/Catalog
Component/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog Component/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                     CatalogComponent tempCatalogComponent = new CatalogComponent();
                     for(int i = 0; i < tempSet.elementContainer.size(); i++)
                     {
                        tempObject = tempSet.elementContainer.get(i);
                        tempCatalogComponent = (CatalogComponent)tempObject;
                        tempCatalogComponent.reInitLists(fileContents_software);
                     }
                  }
                  //condition

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
                  {
                     tempObject = tabManager.get(workareaTabbedPane.
                                          getSelectedIndex());
                     tempSet = (ScenarioElementSet)tempObject;
                     tempSet.setelementSetName(setSaveAsFileName+".BMS");
                     feedbackTextArea.append("Request to save the Condition set " + tempSet.getelementSetName()
+ " is proccessing...\n");
                     try
                     {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.
                        FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
```

```
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                        feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                     }
                     //dac group
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".DMS");
                        feedbackTextArea.append("Request to save the DAC Group set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets Library/DAC
Group/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                }
                //department

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                {
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempSet.setelementSetName(setSaveAsFileName+".EMS");
                    feedbackTextArea.append("Request to save the Department set " +
tempSet.getelementSetName() + " is proccessing...\n");
                    try
                    {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.
                        FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this,
                                "Exception during close",
                                "Exception", JOptionPane.ERROR_MESSAGE);
```

```
                        }
                        initLibraryTree();
                     }
                     //filter
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".FMS");
                        feedbackTextArea.append("Request to save the Filter set " + tempSet.getelementSetName() + "
is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        Filter tempFilter = new Filter();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempFilter = (Filter)tempObject;
                           tempFilter.reInitLists(fileContents_filterapps);

tempFilter.reInitListeners(datapath,startupScenario,fileContents_filterblocking,fileContents_filterapps);
                        }
                     }
                     //integrity

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
```

```
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempSet.setelementSetName(setSaveAsFileName+".IMS");
                       feedbackTextArea.append("Request t o save the Integrity set " + tempSet.getelementSetName() +
" is proccessing...\n");
                       try
                       {
                          feedbackTextArea.append("Attempting to open file...\n");
                          output = new java.io.ObjectOutputStream(new java.io.
                          FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity/"+tempSet.getelementSetName()));
                          feedbackTextArea.append("Save target is: " + datapath + "CyberCIEG E/SDT/Reusable Sets
Library/Integrity/"+tempSet.getelementSetName()+"\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                       }
                       try
                       {
                          feedbackTextArea.append("Attempting to write file...\n");
                          output.writeObject(tempSet);
                          output.flush();
                          feedbackTextArea.append("File written.\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
                       try
                       {
                          feedbackTextArea.append("Attempting to close file...\n");
                          output.close();
                          feedbackTextArea.append("File closed.\n");
                       }
                       catch(IOException ioException)
                       {
                          JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
                       initLibraryTree();
                    }
                    //network

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                    {
                       tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                       tempSet = (ScenarioElementSet)tempObject;
                       tempSet.setelementSetName(setSaveAsFileName+".NMS");
                       feedbackTextArea.append("Request to save the Network set " + tempSet.getelementSetName() +
" is proccessing...\n");
                       try
                       {
                          feedbackTextArea.append("Attempting to open file...\n");
                          output = new java.io.ObjectOutputStream(new java.io.
                          FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network/"+tempSet.getelementSetName()));
                          feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network/"+tempSet.getelementSetName()+"\n");
```

```
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                     }
                     //physical component
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.
                                                    getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".PMS");
                        feedbackTextArea.append("Request to save the Physical "+"Component set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets Library/Physical
Component/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
```

```
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        PhysicalComponent tempPhysicalComponent = new PhysicalComponent();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempPhysicalComponent = (PhysicalComponent)tempObject;
                           tempPhysicalComponent.reInitLists(fileContents_software);

tempPhysicalComponent.reInitListeners(datapath,startupScenario,fileContents_basecomp,fileContents_os,fileContents
_passcomplexity,fileContents_passlength,fileContents_software);
                        }
                     }
                     //procedural settings

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural Settings"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.
                                          getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".LMS");
                        feedbackTextArea.append("Request to save the Procedural "+"Settings set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural
Settings/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                        tempObject = tempSet.elementContainer.get(i);
                        tempProceduralSettings = (ProceduralSettings)tempObject;
                        tempProceduralSettings.reInitLists();
                        tempProceduralSettings.reInitListeners(datapath,startupScenario,
fileContents_passcomplexity, fileContents_passlength, fileContents_passchangefreq);
                    }
                }
                //secrecy

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
                {
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    tempSet.setelementSetName(setSaveAsFileName+".SMS");
                    feedbackTextArea.append("Request to save the Secrecy set " + tempSet.getelementSetName() +
" is proccessing...\n");
                    try
                    {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.
                        FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                    feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
```

```
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                  }
                  //trigger

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                  {
                     tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                     tempSet = (ScenarioElementSet)tempObject;
                     tempSet.setelementSetName(setSaveAsFileName+".TMS");
                     feedbackTextArea.append("Request to save the Trigger set " + tempSet.getelementSetName() +
" is proccessing...\n");
                     try
                     {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.
                        FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                  }
                  //user
                  if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
                  {
```

```
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".UMS");
                        feedbackTextArea.append("Request to save the User set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        User tempUser = new User();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempUser = (User)tempObject;
                           tempUser.reInitLists();
                           tempUser.reInitListeners(datapath, startupScenario);
                        }
                     }
                     //workspace

if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Workspace"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".WMS");
                        feedbackTextArea.append("Request to save the Workspace set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
```

```
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        Workspace tempWorkspace = new Workspace();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempWorkspace = (Workspace)tempObject;
                           tempWorkspace.reInitLists();
                        }
                     }
                     //zone
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".ZMS");
                        feedbackTextArea.append("Request to save the Zone set " +
                        tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone/"+tempSet.getelementSetName()+"\n");
                        }
```

```
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        Zone tempZone = new Zone();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempZone = (Zone)tempObject;
                           tempZone.reInitLists();
                           tempZone.reInitListeners(datapath, startupScenario);
                        }
                     }
                     //component network connection
                     if(workareaTabbedPane.getTitleAt(workareaTabbedPane.
                     getSelectedIndex()).startsWith("Component Network Connection"))
                     {
                        tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                        tempSet = (ScenarioElementSet)tempObject;
                        tempSet.setelementSetName(setSaveAsFileName+".KMS");
                        feedbackTextArea.append("Request to save the Component Network Connection set " +
                           tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.
                           FileOutputStream(datapath + "CyberCIEGE/SDT/Reusable Sets Library/Other/Component
Network Connection/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        try
```

```
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        NetworkConnection tempNetworkConnection = new NetworkConnection();
                        for(int i = 0; i < tempSet.elementContainer.size(); i++)
                        {
                           tempObject = tempSet.elementContainer.get(i);
                           tempNetworkConnection = (NetworkConnection)tempObject;
                           tempNetworkConnection.reInitLists();
                           tempNetworkConnection.reInitListeners(datapath, startupScenario);
                        }
                     }
                  }
                  else
                  {
                     JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                     saveasMenuItemActionPerformed(null);
                  }
                  //END OF SAVE AS
               }
            //saveAllMenuItemActionPerformed()
               //this method causes ALL objects in the tab manager to be saved
               //based on the descriptor type as determiend by the name of the selected tab
               //get the object associtaed with the active tab from the tab manager
               //update the feedback area
               //create an output object to write to
               //write the file
               //reinit any lists and listeners that stopped working as a result of the
               //save (do this as needed).
               private void saveallMenuItemActionPerformed(java.awt.event.ActionEvent evt)
               {
                  int tabCount = workareaTabbedPane.getTabCount();
                  Object tempObject = new Object();
                  Scenario tempScenario;
                  ScenarioElementSet tempSet = new ScenarioElementSet();
                  for(int i = 0; i < tabCount; i++)
                  {
                     //scenario
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Scenario"))
                     {
                        tempObject = tabManager.get(i);
```

```
                        tempScenario = (Scenario)tempObject;
                        feedbackTextArea.append("Request to save the Scenario " + tempScenario.getScenarioFileName()
+ " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()));
                           feedbackTextArea.append("Save target is: " + datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempScenario);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("scenario write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                     }
                     //asset
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Asset"))
                     {
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the Asset set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Asset/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
```

```
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("catalog write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        Asset tempAsset = new Asset();
                        for(int j = 0; j < tempSet.elementContainer.size(); j++)
                        {
                           tempObject = tempSet.elementContainer.get(j);
                           tempAsset = (Asset)tempObject;
                           tempAsset.reInitLists();
                           tempAsset.reInitListeners(datapath, startupScenario);
                        }
                     }
                     //asset goal
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Goal"))
                     {
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the Goal set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Goal/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
```

```
                        System.out.println("catalog write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                     AssetGoal tempAssetGoal = new AssetGoal();
                     for(int j = 0; j < tempSet.elementContainer.size(); j++)
                     {
                        tempObject = tempSet.elementContainer.get(j);
                        tempAssetGoal = (AssetGoal)tempObject;
                        tempAssetGoal.reInitLists();
                        tempAssetGoal.reInitListeners(datapath, startupScenario);
                     }
                  }
                  //catalog component
                  if(workareaTabbedPane.getTitleAt(i).startsWith("Catalog Component"))
                  {
                     tempObject = tabManager.get(i);
                     tempSet = (ScenarioElementSet)tempObject;
                     feedbackTextArea.append("Request to save the Catalog Component set " +
tempSet.getelementSetName() + " is proccessing...\n");
                     try
                     {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Catalog Component/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog Component/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        System.out.println("catalog write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
```

```
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    CatalogComponent tempCatalogComponent = new CatalogComponent();
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObject = tempSet.elementContainer.get(j);
                       tempCatalogComponent = (CatalogComponent)tempObject;
                       tempCatalogComponent.reInitLists(fileContents_software);
                    }
                 }
                 //condition
                 if(workareaTabbedPane.getTitleAt(i).startsWith("Condition"))
                 {
                    tempObject = tabManager.get(i);
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Condition set " + tempSet.getelementSetName() +
" is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapat h +
"CyberCIEGE/SDT/Reusable Sets Library/Condition/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempSet);
                       output.flush();
                       feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                       System.out.println("catalog write exception=" + ioException);
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                 }
                 //dac group
                 if(workareaTabbedPane.getTitleAt(i).startsWith("DAC Group"))
                 {
```

```
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the DAC Group set " + tempSet.getelementSetName()
+ " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("catalog write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                     }
                     //department
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Department"))
                     {
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the Department set " + tempSet.getelementSetName()
+ " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Department/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
```

```
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("catalog write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                     }
                     //filter
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Filter"))
                     {
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the Filter set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Filter/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("catalog write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
```

```
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    Filter tempFilter = new Filter();
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObject = tempSet.elementContainer.get(j);
                       tempFilter = (Filter)tempObject;
                       tempFilter.reInitLists(fileContents_filterapps);
                       tempFilter.reInitListeners(datapath,startupScenario,
                             fileContents_filterblocking,fileContents_filterapps);
                    }
                 }
                 //integrity
                 if(workareaTabbedPane.getTitleAt(i).startsWith("Integrity")) {
                    tempObject = tabManager.get(i);
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Integrity set " + tempSet.getelementSetName() + "
is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempSet);
                       output.flush();
                       feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                       System.out.println("integrity write exception=" + ioException);
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                 }
                 //network
```

```
                    if(workareaTabbedPane.getTitleAt(i).startsWith("Network"))
                    {
                       tempObject = tabManager.get(i);
                       tempSet = (ScenarioElementSet)tempObject;
                       feedbackTextArea.append("Request to save the Network set " + tempSet.getelementSetName() + "
is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Network/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("network write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                       initLibraryTree();
                    }
                    //physical component
                    if(workareaTabbedPane.getTitleAt(i).startsWith("Physical Component")) {
                       tempObject = tabManager.get(i);
                       tempSet = (ScenarioElementSet)tempObject;
                       feedbackTextArea.append("Request to save the Physical Component set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Physical Component/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        System.out.println("physical comp write exception=" + ioException);
                        ioException.printStackTrace();
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    PhysicalComponent tempPhysicalComponent = new PhysicalComponent();
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObject = tempSet.elementContainer.get(j);
                        tempPhysicalComponent = (PhysicalComponent)tempObject;
                        tempPhysicalComponent.reInitLists(fileContents_software);
                        tempPhysicalComponent.reInitListeners(datapath,startupScenario,
                                fileContents_basecomp,fileContents_os,
                                fileContents_passcomplexity,
                                fileContents_passlength,fileContents_software);
                    }
                }
                //procedural settings
                if(workareaTabbedPane.getTitleAt(i).startsWith("Procedural Settings"))
                {
                   tempObject = tabManager.get(i);
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Procedural Settings set " +
tempSet.getelementSetName() + " is proccessing...\n");
                    try
                    {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural Settings/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
```

```
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        System.out.println("catalog write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                     ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                     for(int j = 0; j < tempSet.elementContainer.size(); j++)
                     {
                        tempObject = tempSet.elementContainer.get(j);
                        tempProceduralSettings = (ProceduralSettings)tempObject;
                        tempProceduralSettings.reInitLists();
                        tempProceduralSettings.reInitListeners(datapath,startupScenario,
                              fileContents_passcomplexity, fileContents_passlength,
                              fileContents_passchangefreq);
                     }
                  }
                  //secrecy
                  if(workareaTabbedPane.getTitleAt(i).startsWith("Secrecy"))
                  {
                     tempObject = tabManager.get(i);
                     tempSet = (ScenarioElementSet)tempObject;
                     feedbackTextArea.append("Request to save the Secrecy set " + tempSet.getelementSetName() + "
is proccessing...\n");
                      try
                      {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
```

```
                        System.out.println("secrecy write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                }
                //trigger
                if(workareaTabbedPane.getTitleAt(i).startsWith("Trigger"))
                {
                    tempObject = tabManager.get(i);
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Trigger set " + tempSet.getelementSetName() + " is
proccessing...\n");
                    try
                    {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Trigger/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        System.out.p rintln("catalog write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                }
```

```
                    //user
                    if(workareaTabbedPane.getTitleAt(i).startsWith("User"))
                    {
                       tempObject = tabManager.get(i);
                       tempSet = (ScenarioElementSet)tempObject;
                       feedbackTextArea.append("Request to save the User set " + tempSet.getelementSetName() + " is
proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempSet.getelementSetName()));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("catalog write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        User tempUser = new User();
                        for(int j = 0; j < tempSet.elementContainer.size(); j++)
                        {
                           tempObject = tempSet.elementContainer.get(j);
                           tempUser = (User)tempObject;
                           tempUser.reInitLists();
                           tempUser.reInitListeners(datapath, startupScenario);
                        }
                    }
                    //workspace
                    if(workareaTabbedPane.getTitleAt(i).startsWith("Workspace"))
                    {
                       tempObject = tabManager.get(i);
                       tempSet = (ScenarioElementSet)tempObject;
                       feedbackTextArea.append("Request to save the Workspace set " + tempSet.getelementSetName() +
" is proccessing...\n");
                        try
                        {
```

```
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Workspace/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Workspace/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        System.out.println("catalog write exception=" + ioException);
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     initLibraryTree();
                     Workspace tempWorkspace = new Workspace();
                     for(int j = 0; j < tempSet.elementContainer.size(); j++)
                     {
                        tempObject = tempSet.elementContainer.get(j);
                        tempWorkspace = (Workspace)tempObject;
                        tempWorkspace.reInitLists();
                     }
                  }
                  //zone
                  if(workareaTabbedPane.getTitleAt(i).startsWith("Zone")) {
                     tempObject = tabManager.get(i);
                     tempSet = (ScenarioElementSet)tempObject;
                     feedbackTextArea.append("Request to save the Zone set " + tempSet.getelementSetName() + " is
proccessing...\n");
                     try
                     {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Zone/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone/"+tempSet.getelementSetName()+"\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
```

```
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
                        {
                           System.out.println("zone write exception=" + ioException);
                           JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try {
                           feedbackTextArea.append("Attempting to close file...\n");
                           output.close();
                           feedbackTextArea.append("File closed.\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        initLibraryTree();
                        Zone tempZone = new Zone();
                        for(int j = 0; j < tempSet.elementContainer.size(); j++)
                        {
                           tempObject = tempSet.elementContainer.get(j);
                           tempZone = (Zone)tempObject;
                           tempZone.reInitLists();
                           tempZone.reInitListeners(datapath, startupScenario);
                        }
                     }
                     //network connection
                     if(workareaTabbedPane.getTitleAt(i).startsWith("Component Network Connection"))
                     {
                        tempObject = tabManager.get(i);
                        tempSet = (ScenarioElementSet)tempObject;
                        feedbackTextArea.append("Request to save the Component Network Connection set " +
tempSet.getelementSetName() + " is proccessing...\n");
                        try
                        {
                           feedbackTextArea.append("Attempting to open file...\n");
                           output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Component Network
Connection/"+tempSet.getelementSetName()+""));
                           feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection/"+tempSet.getelementSetName()+"\n");
                        }
                        catch(IOException ioException)
                        {
                           JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                        }
                        try
                        {
                           feedbackTextArea.append("Attempting to write file...\n");
                           output.writeObject(tempSet);
                           output.flush();
                           feedbackTextArea.append("File written.\n");
                        }
                        catch(IOException ioException)
```

```
                    {
                       System.out.println("catalog write exception=" + ioException);
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    NetworkConnection tempNetworkConnection = new NetworkConnection();
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObject = tempSet.elementContainer.get(j);
                       tempNetworkConnection = (NetworkConnection)tempObject;
                       tempNetworkConnection.reInitLists();
                       tempNetworkConnection.reInitListeners(datapath, startupScenario);
                    }
                 }
              }
           }
        //savescenarioMenuItemActionPerformed()
           //this method allows the user to save the current scenario
           //if the selected tab is the scenario tab
           //then it checks for a legal name
           //then it creates an output object to write the file to
           private void savescenarioMenuItemActionPerformed(java.awt.event.ActionEvent evt)
           {
              Object tempObject = new Object();
              Scenario tempScenario;
              if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Scenario"))
              {
                 tempObject = tabManager.get(0);
                 tempScenario = (Scenario)tempObject;
                 if(tempScenario.getScenarioFileName() != null && tempScenario.getScenarioFileName() != " " &&
tempScenario.getScenarioFileName().length() != 0)
                 {
                    feedbackTextArea.append("Request to save the Scenario " + tempScenario.getScenarioFileName()
+ " is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()));
                       feedbackTextArea.append("Save target is: " + datapath +
"CyberCIEGE/SDT/Scenarios/"+tempScenario.getScenarioFileName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempScenario);
```

```
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                     try
                     {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                     }
                     catch(IOException ioException)
                     {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                     }
                  }
                  else
                  {
                     JOptionPane.showMessageDialog(this, "Unable to save - Scenario File Name is required",
"Scenario File Name Error", JOptionPane.ERROR_MESSAGE);
                     tempScenario.setScenarioFileName(JOptionPane.showInputDialog("Enter the scenario file
name."));
                     savescenarioMenuItemActionPerformed(null);
                  }
               }
            }
         //saveMenuItemActionPerformed()
           //this method allows the scenario element set in the currently selected
           //tab to be saved
           //based on the descriptor type as determiend by the name of the selected tab
           //get the object associtaed with the active tab from the tab manager
           //set the sets name to the new name
           //update the feedback area
           //create an output object to write to
           //write the file
           //reinit any lists and listeners that stopped working as a result of the
           //save (do this as needed).
           private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt)
           {
              Object tempObject = new Object();
              ScenarioElementSet tempSet = new ScenarioElementSet();
              //asset
              if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Asset"))
              {
                 tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                 tempSet = (ScenarioElementSet)tempObject;
                 feedbackTextArea.append("Request to save the Asset set " + tempSet.getelementSetName() + " is
proccessing...\n");
                  try
                  {
                     feedbackTextArea.append("Attempting to open file...\n");
                     output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Asset/"+tempSet.getelementSetName()));
                     feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Asset/"+tempSet.getelementSetName()+"\n");
                  }
                  catch(IOException ioException)
                  {
```

```
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    ioException.printStackTrace();
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                Asset tempAsset = new Asset();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                    tempObject = tempSet.elementContainer.get(i);
                    tempAsset = (Asset)tempObject;
                    tempAsset.reInitLists();
                    tempAsset.reInitListeners(datapath, startupScenario);
                }
            }
            //asset goal
            if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Goal"))
            {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Goal set " + tempSet.getelementSetName() + " is
proccessing...\n");
                try
                {
                    feedbackTextArea.append("Attempting to open file...\n");
                    output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Goal/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Goal/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
```

```
                    }
                    catch(IOException ioException)
                    {
                       ioException.printStackTrace();
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    AssetGoal tempAssetGoal = new AssetGoal();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempAssetGoal = (AssetGoal)tempObject;
                       tempAssetGoal.reInitLists();
                       tempAssetGoal.reInitListeners(datapath, startupScenario);
                    }
                 }
                 //catalog component
                 if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Catalog
Component"))
                 {
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Catalog Component set "+
tempSet.getelementSetName() + " is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable "+"Sets Library/Catalog Component/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Catalog "+"Component/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempSet);
                       output.flush();
                       feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
```

```
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                 }
                 initLibraryTree();
                 CatalogComponent tempCatalogComponent = new CatalogComponent();
                 for(int i = 0; i < tempSet.elementContainer.size(); i++)
                 {
                    tempObject = tempSet.elementContainer.get(i);
                    tempCatalogComponent = (CatalogComponent)tempObject;
                    tempCatalogComponent.reInitLists(fileContents_software);
                 }
              }
              //condition
              if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Condition"))
              {
                 tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                 tempSet = (ScenarioElementSet)tempObject;
                 feedbackTextArea.append("Request to save the Condition set " + tempSet.getelementSetName() + " is
proccessing...\n");
                 try
                 {
                    feedbackTextArea.append("Attempting to open file...\n");
                    output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Condition/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Condition/"+tempSet.getelementSetName()+"\n");
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                 }
                 try
                 {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                 }
                 try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                 }
                 catch(IOException ioException)
                 {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                 }
                 initLibraryTree();
              }
              //dac group
```

```
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("DAC
Group"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the DAC Group set " + tempSet.getelementSetName() + "
is proccessing...\n");
                   try
                   {
                      feedbackTextArea.append("Attempting to open file...\n");
                      output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempSet.getelementSetName()));
                      feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/DAC Group/"+tempSet.getelementSetName()+"\n");
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      feedbackTextArea.append("Attempting to write file...\n");
                      output.writeObject(tempSet);
                      output.flush();
                      feedbackTextArea.append("File written.\n");
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_M ESSAGE);
                   }
                   try {
                      feedbackTextArea.append("Attempting to close file...\n");
                      output.close();
                      feedbackTextArea.append("File closed.\n");
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                   }
                   initLibraryTree();
                }
                //department
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Department"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the Department set " + tempSet.getelementSetName() + "
is proccessing...\n");
                   try
                   {
                      feedbackTextArea.append("Attempting to open file...\n");
                      output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Department/"+tempSet.getelementSetName()));
                      feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Department/"+tempSet.getelementSetName()+"\n");
                   }
                   catch(IOException ioException)
                   {
```

```
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
            }
            //filter
            if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Filter"))
            {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Filter set " + tempSet.getelementSetName() + " is
proccessing...\n");
                try
                {
                    feedbackTextArea.append("Attempting to open file...\n");
                    output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Filter/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Filter/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
```

```
                    feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                Filter tempFilter = new Filter();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                   tempObject = tempSet.elementContainer.get(i);
                   tempFilter = (Filter)tempObject;
                   tempFilter.reInitLists(fileContents_filterapps);
                   tempFilter.reInitListeners(datapath,startupScenario,
                         fileContents_filterblocking,fileContents_filterapps);
                }
             }
             //integrity
             if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Integrity"))
             {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Integrity set " + tempSet.getelementSetName() + " is
proccessing...\n");
                try
                {
                   feedbackTextArea.append("Attempting to open file...\n");
                   output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempSet.getelementSetName()));
                   feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Integrity/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   feedbackTextArea.append("Attempting to write file...\n");
                   output.writeObject(tempSet);
                   output.flush();
                   feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                   feedbackTextArea.append("Attempting to close file...\n");
                   output.close();
                   feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
             }
```

```
                //network
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Network"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the Network set " + tempSet.getelementSetName() + " is
proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Network/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Network/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempSet);
                       output.flush();
                       feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                   initLibraryTree();
                }
                //physical component
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Physical
Component"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the Physical Component set " +
tempSet.getelementSetName() + " is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Physical Component/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Physical Component/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
```

```
                {
                   JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   feedbackTextArea.append("Attempting to write file...\n");
                   output.writeObject(tempSet);
                   output.flush();
                   feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                   feedbackTextArea.append("Attempting to close file...\n");
                   output.close();
                   feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                PhysicalComponent tempPhysicalComponent = new PhysicalComponent();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                   tempObject = tempSet.elementContainer.get(i);
                   tempPhysicalComponent = (PhysicalComponent)tempObject;
                   tempPhysicalComponent.reInitLists(fileContents_software);
                   tempPhysicalComponent.reInitListeners(datapath,startupScenario,
                            fileContents_basecomp,fileContents_os,
                            fileContents_passcomplexity,
                            fileContents_passlength,fileContents_software);
                }
             }
             //procedural settings
             if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Procedural
Settings"))
             {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Procedural Settings set " +
tempSet.getelementSetName() + " is proccessing...\n");
                try
                {
                   feedbackTextArea.append("Attempting to open file...\n");
                   output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural Settings/"+tempSet.getelementSetName()));
                   feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Procedural Settings/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
```

```
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackText Area.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                ProceduralSettings tempProceduralSettings = new ProceduralSettings();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                   tempObject = tempSet.elementContainer.get(i);
                   tempProceduralSettings = (ProceduralSettings)tempObject;
                   tempProceduralSettings.reInitLists();
                   tempProceduralSettings.reInitListeners(datapath,startupScenario,
                         fileContents_passcomplexity, fileContents_passlength,
                         fileContents_passchangefreq);
                }
             }
             //secrecy
             if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Secrecy"))
             {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Secrecy set " + tempSet.getelementSetName() + " is
proccessing...\n");
                try
                {
                    feedbackTextArea.append("Attempting to open file...\n");
                   output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Secrecy/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
```

```
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                }
                //trigger
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Trigger"))
                {
                    tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                    tempSet = (ScenarioElementSet)tempObject;
                    feedbackTextArea.append("Request to save the Trigger set " + tempSet.getelementSetName() + " is
proccessing...\n");
                    try
                    {
                        feedbackTextArea.append("Attempting to open file...\n");
                        output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Trigger/"+tempSet.getelementSetName()));
                        feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Trigger/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                        feedbackTextArea.append("Attempting to write file...\n");
                        output.writeObject(tempSet);
                        output.flush();
                        feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                        feedbackTextArea.append("Attempting to close file...\n");
                        output.close();
                        feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                }
                //user
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("User"))
```

```
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the User set " + tempSet.getelementSetName() + " is
proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempSet.getelementSetName()));
                       feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/User/"+tempSet.getelementSetName()+"\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       feedbackTextArea.append("Attempting to write file...\n");
                       output.writeObject(tempSet);
                       output.flush();
                       feedbackTextArea.append("File written.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    User tempUser = new User();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempUser = (User)tempObject;
                       tempUser.reInitLists();
                       tempUser.reInitListeners(datapath, startupScenario);
                    }
                }
                //workspace
                if(workareaTabbedPane.getTitleAt(workareaTabbedPane.
                            getSelectedIndex()).startsWith("Workspace"))
                {
                   tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                   tempSet = (ScenarioElementSet)tempObject;
                   feedbackTextArea.append("Request to save the Workspace set " +
                           tempSet.getelementSetName() + " is proccessing...\n");
                    try
                    {
                       feedbackTextArea.append("Attempting to open file...\n");
                       output = new java.io.ObjectOutputStream(new java.io.
```

```
                        FileOutputStream(datapath +
                        "CyberCIEGE/SDT/Reusable Sets Library/Workspace/"+
                        tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath +
                        "CyberCIEGE/SDT/Reusable Sets Library/Workspace/"+
                        tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                        "Exception during output stream creation",
                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during writing",
                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close",
                        "Exception", JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                Workspace tempWorkspace = new Workspace();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                    tempObject = tempSet.elementContainer.get(i);
                    tempWorkspace = (Workspace)tempObject;
                    tempWorkspace.reInitLists();
                }
            }
            //zone
            if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Zone"))
            {
                tempObject = tabManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Zone set " + tempSet.getelementSetName() + " is
proccessing...\n");
                try
                {
                    feedbackTextArea.append("Attempting to open file...\n");
                    output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Zone/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Zone/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
```

```
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try {
                    feedbackTextArea.append("Attempting to close file...\n");
                    output.close();
                    feedbackTextArea.append("File closed.\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                initLibraryTree();
                Zone tempZone = new Zone();
                for(int i = 0; i < tempSet.elementContainer.size(); i++)
                {
                   tempObject = tempSet.elementContainer.get(i);
                    tempZone = (Zone)tempObject;
                    tempZone.reInitLists();
                    tempZone.reInitListeners(datapath, startupScenario);
                }
            }
            //component network connection
            if(workareaTabbedPane.getTitleAt(workareaTabbedPane.getSelectedIndex()).startsWith("Component
Network Connection"))
            {
                tempObject = t abManager.get(workareaTabbedPane.getSelectedIndex());
                tempSet = (ScenarioElementSet)tempObject;
                feedbackTextArea.append("Request to save the Component Network Connection set " +
tempSet.getelementSetName() + " is proccessing...\n");
                try
                {
                    feedbackTextArea.append("Attempting to open file...\n");
                    output = new java.io.ObjectOutputStream(new java.io.FileOutputStream(datapath +
"CyberCIEGE/SDT/Reusable Sets Library/Other/Component Network Connection/"+tempSet.getelementSetName()));
                    feedbackTextArea.append("Save target is: " + datapath + "CyberCIEGE/SDT/Reusable Sets
Library/Other/Component Network Connection/"+tempSet.getelementSetName()+"\n");
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during output stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    feedbackTextArea.append("Attempting to write file...\n");
                    output.writeObject(tempSet);
                    output.flush();
                    feedbackTextArea.append("File written.\n");
```

```
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during writing", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    try {
                       feedbackTextArea.append("Attempting to close file...\n");
                       output.close();
                       feedbackTextArea.append("File closed.\n");
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during close", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    initLibraryTree();
                    NetworkConnection tempNetworkConnection = new NetworkConnection();
                    for(int i = 0; i < tempSet.elementContainer.size(); i++)
                    {
                       tempObject = tempSet.elementContainer.get(i);
                       tempNetworkConnection = (NetworkConnection)tempObject;
                       tempNetworkConnection.reInitLists();
                       tempNetworkConnection.reInitListeners(datapath, startupScenario);
                    }
                 }
              }
           //networkMenuItemActionPerformed()
              //- creates a new network object
              //- creates a new scenario element set
              //- prompts for a name for the new set
              //- tests the name provided to make sure it is legal
              //- appends the file extension for a network .NMS to the file
              //name
              //- adds the network object to the scenario element set
              //- creates a new tab in the tabbed work area, names it and adds the
              //new object to it
              //- if the file name provided was not legal the method prompts for
              //input until it is legal
              private void networkMenuItemActionPerformed(java.awt.event.ActionEvent evt)
              {
                 Network newNetwork = new Network();
                 ScenarioElementSet tempnetworkset = new ScenarioElementSet();
                 tempnetworkset.setelementSetName(JOptionPane.showInputDialog("Enter the network file name."));

                 if(tempnetworkset.getelementSetName() != null && tempnetworkset.getelementSetName() != " " &&
tempnetworkset.getelementSetName().length() != 0)
                 {
                    tempnetworkset.setelementSetName(tempnetworkset.getelementSetName()+".NMS");
                    tempnetworkset.setelementType("Network");
                    tempnetworkset.elementContainer.add(newNetwork);

workareaTabbedPane.addTab("Network:"+tempnetworkset.getelementSetName(),(Network)tempnetworkset.elementC
ontainer.lastElement());
                    tabManager.insertElementAt(tempnetworkset, (workareaTabbedPane.getTabCount()-1));
                 }
                 else
                 {
                    JOptionPane.showMessageDialog(this, "A file name is required", "File Name Error",
JOptionPane.ERROR_MESSAGE);
                    networkMenuItemActionPerformed(null);
                 }
```

```
    }

    /** Exit the Application */
    private void exitForm(java.awt.event.WindowEvent evt)
   {
     System.exit(0);
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
       try
       {
          UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
       }
       catch (Exception e)
       {
          JOptionPane.showMessageDialog(null, "Unable to load the Windows Look and Feel interface
defaulting to Java Look and Feel.", "Exception", JOptionPane.ERROR_MESSAGE);
       }
       //the show method is inherited - it is not presented in these files
       new ScenarioDefinitionTool().show();
    }

    //this method is used to create the fileNode object that is responsible for
    //populating the reusable sets library tree
    //for details on the fileNode see FileNode.java
    private DefaultTreeModel librarytreePopulate()
    {
       File root = new File(datapath + "CyberCIEGE/SDT/Reusable Sets Library");
       FileNode rootNode = new FileNode(root), node;
       rootNode.explore();
       return new DefaultTreeModel(rootNode);
    }

    //this method is used to create the objectNode that is responsible for
    //populating the contents of the scenario tree
    //for details on the objectNode see ObjectNode.java
    public DefaultTreeModel scenariotreePopulate()
    {
      Object tempObject = tabManager.get(0);
       Scenario tempScenario = (Scenario)tempObject;
       ObjectNode rootNode = new ObjectNode(tempScenario);
       return rootNode.explore();
    }

    //this method is used to force an update of the reusable sets library tree
    public void initLibraryTree()
    {
       libraryTree =  new javax.swing.JTree(librarytreePopulate());

libraryTree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
       DragSource libraryTreeDragSource = DragSource.getDefaultDragSource();
       libraryTreeDragSource.createDefaultDragGestureRecognizer(libraryTree,
DnDConstants.ACTION_COPY,
       new DragGestureListener()
       {
          public void dragGestureRecognized(DragGestureEvent event)
          {
             JTree tempTree = (JTree)event.getComponent();
             String tempString = tempTree.getSelectionPath().toString();
```

```
                    ScenarioElementSet draggedValue = getDragSourceFileFromTree(tempString);
                    Transferable transferable = new ScenarioElementSetTransferable(draggedValue);
                    event.startDrag(null,transferable,new ScenarioElementSetDragSourceListener());
                }
            });
            libraryScrollPane.setPreferredSize(new java.awt.Dimension(175, 363));
            libraryTree.addMouseListener(new java.awt.event.MouseAdapter()
            {
                public void mousePressed(java.awt.event.MouseEvent evt)
                {
                    libraryTreeMousePressed(evt);
                }
            });
            libraryTree.addTreeExpansionListener(new javax.swing.event.TreeExpansionListener()
            {
                public void treeCollapsed(javax.swing.event.TreeExpansionEvent evt)
                {
                }
                public void treeExpanded(javax.swing.event.TreeExpansionEvent evt)
                {
                    libraryTreeTreeExpanded(evt);
                }
            });
            libraryScrollPane.setViewportView(libraryTree);
            treePanel.add(scenarioScrollPane);
            libraryTree.repaint();
        }

        //this method is used to force an update of the scenario tree
        public void initScenarioTree()
        {
            scenarioTree = new javax.swing.JTree(scenariotreePopulate());

scenarioTree.getSelectionModel().setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
            scenarioTreeTarget = new DropTarget(scenarioTree,new
ScenarioElementSetDropTargetListener(startupScenario, scenarioTree));
            scenarioTree.addMouseListener(new java.awt.event.MouseAdapter()
            {
                public void mousePressed(java.awt.event.MouseEvent evt)
                {
                    scenarioTreeMousePressed(evt);
                }
            });
            scenarioScrollPane.setViewportView(scenarioTree);
            treePanel.add(scenarioScrollPane);
            scenarioTree.repaint();
        }

        //this method is used to turn a string into an object
        private Object makeObj(final String item)
        {
            return new Object()
            {
                public String toString()
                {
                    return item;
                }
            };
        }

        //this method reads each of the individual .ini files that are stored in
        //C:/Program Files/CyberCIEGE/
```

```
        //ALL of the ini files are nothing more that formatted text files
        //they serve the purpose of providing static game information to the
        //SDT
        //each .ini file has a vector associated with it.  the vector names start
        //with fileContents_
        //The vectors store a single line entry from the file in each index
        //The files for condition and trigger hold one complete condition/trigger
        //entry per line.  They are the only files set up in this manner.
        public void readINIFiles()
        {
           //get input from static files
           //base components
           //filterblocking
           //filterapplications
           //datapath
           //floorplan
           //OS
           //password complexity
           //password length
           //password chagnge freq
           //software
           //background check
           //organization
           String fromFile;
           fileContents_filterblocking = new java.util.Vector();
           fileContents_floorplan = new java.util.Vector();
           fileContents_filterapps = new java.util.Vector();
           fileContents_basecomp = new java.util.Vector();
           datapath = new String();
           datapath = "C:/Program Files/";
           fileContents_os = new java.util.Vector();
           fileContents_passcomplexity = new java.util.Vector();
           fileContents_passlength = new java.util.Vector();
           fileContents_passchangefreq =  new java.util.Vector();
           fileContents_software = new java.util.Vector();
           fileContents_backgroundcheck = new java.util.Vector();
           fileContents_organization = new java.util.Vector();
           fileContents_condition = new java.util.Vector();
           fileContents_trigger = new java.util.Vector();
           BufferedReader in;

           try
           {
                 //datapath
                 in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/datapath.ini"));

                 try
                 {
                       while((fromFile = in.readLine()) != null)
                       {
        //                   System.out.println("From file: " + fromFile);//this works!!!
                          datapath = fromFile;
                       }
                       try
                       {
                             in.close();
                       }
                       catch(IOException ioException)
                       {
                             JOptionPane.showMessageDialog(null, "File Close Exception - Datapath", "Exception",
JOptionPane.ERROR_MESSAGE);
                       }
```

```
                    catch(Exception exception)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Datapath", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - Datapath", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                //background check
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/backgroundcheck.ini"));

                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
        //              System.out.println("From file: " + fromFile);//this works!!!
                       fileContents_backgroundcheck.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - background check",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(Exception exception)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - background check",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - background check",
"Exception", JOptionPane.ERROR_MESSAGE);
                }
                //filterblocking
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/filterblocking.ini"));

                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
        //              System.out.println("From file: " + fromFile);//this works!!!
                       fileContents_filterblocking.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Filter blocking",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(Exception exception)
                    {
```

```
                                JOptionPane.showMessageDialog(null, "File Close Exception - Filter blocking",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Read Exception - Filter blocking", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    //filterapps
                    in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/filterapplications.ini"));

                    try
                    {
                        while((fromFile = in.readLine()) != null)
                        {
            //                System.out.println("From file: " + fromFile);//this works!!!
                           fileContents_filterapps.add(fromFile);
                        }
                        try
                        {
                            in.close();
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(null, "File Close Exception - Filter applications",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(Exception exception)
                        {
                            JOptionPane.showMessageDialog(null, "File Close Exception - Filter applications",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Read Exception - Filter applications",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    //base component
                    in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/basecomponent.ini"));

                    try
                    {
                        while((fromFile = in.readLine()) != null)
                        {
            //                System.out.println("From file: " + fromFile);//this works!!!
                           fileContents_basecomp.add(fromFile);
                        }
                        try
                        {
                            in.close();
                        }
                        catch(IOException ioException)
                        {
                            JOptionPane.showMessageDialog(null, "File Close Exception - Base Component",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }
                        catch(Exception exception)
                        {
                            JOptionPane.showMessageDialog(null, "File Close Exception - Base Component",
"Exception", JOptionPane.ERROR_MESSAGE);
```

```
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - Base Component",
"Exception", JOptionPane.ERROR_MESSAGE);
                }

                //floorplan
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/floorplan.ini"));

                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
        //              System.out.println("From file: " + fromFile);//this works!!!
                       fileContents_floorplan.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Floorplan", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                    catch(Exception exception)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Base Component",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - Floorplan", "Exception",
JOptionPane.ERROR_MESSAGE);
                }

                //OS
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/operatingsystems.ini"));
                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
                       fileContents_os.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - OS", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - OS", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
```

```
                        //Password Length
                        in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/passwordlength.ini"));
                        try
                        {
                              while((fromFile = in.readLine()) != null)
                              {
                                 fileContents_passlength.add(fromFile);
                              }
                              try
                              {
                                    in.close();
                              }
                              catch(IOException ioException)
                              {
                                    JOptionPane.showMessageDialog(null, "File Close Exception - Password Length",
"Exception", JOptionPane.ERROR_MESSAGE);
                              }
                        }
                        catch(IOException ioException)
                        {
                              JOptionPane.showMessageDialog(null, "File Read Exception - Password Length",
"Exception", JOptionPane.ERROR_MESSAGE);
                        }

                        //Password Complexity
                        in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/passwordcomplexity.ini"));
                        try
                        {
                              while((fromFile = in.readLine()) != null)
                              {
                                 fileContents_passcomplexity.add(fromFile);
                              }
                              try
                              {
                                    in.close();
                              }
                              catch(IOException ioException)
                              {
                                    JOptionPane.showMessageDialog(null, "File Close Exception - Password Complexity",
"Exception", JOptionPane.ERROR_MESSAGE);
                              }
                        }
                        catch(IOException ioException)
                        {
                              JOptionPane.showMessageDialog(null, "File Read Exception - Password Complexity",
"Exception", JOptionPane.ERROR_MESSAGE);
            //                 System.out.println("pwdcomplex close exception=" + ioException);
                        }
                        //Password Change Frequency
                        in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/passwordchangefreq.ini"));
                        try
                        {
                              while((fromFile = in.readLine()) != null)
                              {
                                 fileContents_passchangefreq.add(fromFile);
                              }
                              try
                              {
                                    in.close();
                              }
                              catch(IOException ioException)
```

```
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Password Change Freq",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - Password Change Freq",
"Exception", JOptionPane.ERROR_MESSAGE);
            //          System.out.println("pwdcomplex close exception=" + ioException);
                }
                //software
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/software.ini"));
                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
                       fileContents_software.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - Software", "Exception",
JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - Software", "Exception",
JOptionPane.ERROR_MESSAGE);
                }

                //organization
                in = new BufferedReader(new FileReader(datapath + "CyberCIEGE/organization.ini"));
                try
                {
                    while((fromFile = in.readLine()) != null)
                    {
                       fileContents_organization.add(fromFile);
                    }
                    try
                    {
                        in.close();
                    }
                    catch(IOException ioException)
                    {
                        JOptionPane.showMessageDialog(null, "File Close Exception - organization",
"Exception", JOptionPane.ERROR_MESSAGE);
                    }
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(null, "File Read Exception - organization", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                //condition
                try
                {
                   FileReader conditionTokens = new FileReader(datapath + "CyberCIEGE/condition.ini");
```

```
   StreamTokenizer cond = new StreamTokenizer(conditionTokens);
   Vector lineVec = new Vector();

   //Prepare the tokenizer for Java-style tokenizing rules
   cond.parseNumbers();
   cond.wordChars('_', '_');
   cond.eolIsSignificant(true);

   //If whitespace is not to be discarded, make this call
   cond.ordinaryChars(0, ' ');

   //These calls caused comments to be discarded
   cond.slashSlashComments(true);
   cond.slashStarComments(true);

   //Parse the file
   int token = 65535;
   while (token != StreamTokenizer.TT_EOF)
   {
      token = cond.nextToken();
      switch (token)
      {
         case StreamTokenizer.TT_NUMBER:
            //A number was found; the value is in nval
            double num = cond.nval;
            break;
         case StreamTokenizer.TT_WORD:
            //A word was found; the value is in sval
            String word = cond.sval;
            lineVec.addElement(word);
            break;
         case '"':
            //A double-quoted string was found;
            //sval contains the contents
            String dquoteVal = cond.sval;
            break;
         case '\'':
            //A single-quoted string was found;
            //sval contains the contents
            String squoteVal = cond.sval;
            break;
         case StreamTokenizer.TT_EOL:
            //End of line character found
            fileContents_condition.addElement(lineVec);
            lineVec = new Vector();
            break;
         case StreamTokenizer.TT_EOF:
            //End of file has been reached
            fileContents_condition.addElement(lineVec);
            break;
         default:
            //A regular character was found;
            //the value is the token itself
            char ch = (char)cond.ttype;
            break;
      }
   }
   conditionTokens.close();
}
catch(IOException ioException)
{
```

```
                    JOptionPane.showMessageDialog(null, "File Close Exception - organization", "Exception",
JOptionPane.ERROR_MESSAGE);
                }


                //trigger
                try
                {
                   FileReader triggerTokens = new FileReader(datapath + "CyberCIEGE/trigger.ini");
                   StreamTokenizer trig = new StreamTokenizer(triggerTokens);
                   Vector lineVec2 = new Vector();

                   //Prepare the tokenizer for Java-style tokenizing rules
                   trig.parseNumbers();
                   trig.wordChars('_', '_');
                   trig.eolIsSignificant(true);

                   //If whitespace is not to be discarded, make this call
                   trig.ordinaryChars(0, ' ');

                   //These calls caused comments to be discarded
                   trig.slashSlashComments(true);
                   trig.slashStarComments(true);

                   //Parse the file
                   int token = 65535;
                   while (token != StreamTokenizer.TT_EOF) {
                      token = trig.nextToken();
                      switch (token) {
                      case StreamTokenizer.TT_NUMBER:
                         //A number was found; the value is in nval
                         double num = trig.nval;
                         break;
                      case StreamTokenizer.TT_WORD:
                         //A word was found; the value is in sval
                         String word = trig.sval;
                         lineVec2.addElement(word);
//                          System.out.println(word);
                         break;
                      case '"':
                         //A double-quoted string was found; sval contains the contents
                         String dquoteVal = trig.sval;
                         break;
                      case '\'':
                         //A single-quoted string was found; sval contains the contents
                         String squoteVal = trig.sval;
                         break;
                      case StreamTokenizer.TT_EOL:
                         //End of line character found
                         fileContents_trigger.addElement(lineVec2);
                         lineVec2 = new Vector();
                         break;
                      case StreamTokenizer.TT_EOF:
                         //End of file has been reached
                         fileContents_trigger.addElement(lineVec2);
                         break;
                      default:
                         //A regular character was found; the value is the token itself
                         char ch = (char)trig.ttype;
                         break;
                      }
                   }
```

```
                    triggerTokens.close();
//                  Object tempObj = new Object();
//                  Vector tempVec = new Vector();
//                  for(int i = 0; i < fileContents_trigger.size(); i++)
//                  {
//                     tempObj = fileContents_trigger.get(i);
//                     tempVec = (Vector)tempObj;
//                     for(int j = 0; j < tempVec.size(); j++)
//                     {
//                        feedbackTextArea.append(tempVec.get(j).toString()+"\n");
//                     }
//                  }
                  }
                  catch(IOException ioException)
                  {
                        JOptionPane.showMessageDialog(null, "File Close Exception - organization", "Exception",
JOptionPane.ERROR_MESSAGE);
                  }
               }
               catch(FileNotFoundException filenotfoundException)
               {
               JOptionPane.showMessageDialog(null, "File Not Found Exception", "Exception",
JOptionPane.ERROR_MESSAGE);
               }
            }

            //this method is part of the drag and drop (DND) operation.
            //its purpose is to take a tree path, convert it to a file path
            //then get the file associated with the path and return it.
            //this method is used when the dragGestureRecognized() method of the
            //library tree fires and a DND operation is started
            private ScenarioElementSet getDragSourceFileFromTree(String aString)
            {
               String filePathPrefix = datapath + "CyberCIEGE/SDT/";
               ScenarioElementSet theFile = new ScenarioElementSet();
               aString = aString.replaceAll("\\[","");
               aString = aString.replaceAll("]","");
               aString = aString.replaceAll(", ", "/");
               aString = aString.trim();
               aString = filePathPrefix + aString;
               File targetFile = new File(aString);
               if(!targetFile.isDirectory())
               {
                  try
                  {
                     input = new java.io.ObjectInputStream(new java.io.FileInputStream(aString));
                  }
                  catch(IOException ioException)
                  {
                     JOptionPane.showMessageDialog(this, "Exception during input stream creation", "Exception",
JOptionPane.ERROR_MESSAGE);
                  }
                  try
                  {
                     theFile = (ScenarioElementSet)input.readObject();
                  }
                  catch(ClassNotFoundException classnotfoundException)
                  {
                     JOptionPane.showMes sageDialog(this, "Exception during file read - the object was not found",
"Exception", JOptionPane.ERROR_MESSAGE);
                  }
                  catch(IOException ioException)
```

```
                {
                   JOptionPane.showMessageDialog(this, "Exception during file read", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                   input.close();
                }
                catch(IOException ioException)
                {
                   JOptionPane.showMessageDialog(this, "Exception during file close", "Exception",
JOptionPane.ERROR_MESSAGE);
                }
             }
             return theFile;
          }

          //this nifty little method takes in a tree path and turns it into a file
          //path
          private String treePathtoFilePath(String aString)
          {
             String filePathPrefix = datapath + "CyberCIEGE/SDT/";
             aString = aString.replaceAll("Current Scenario: .*CSM, ","Reusable Sets Library/");
             aString = aString.replaceAll("\\[","");
             aString = aString.replaceAll("]","");
             aString = aString.replaceAll(", ", "/");
             aString = aString.trim();
             aString = filePathPrefix + aString;
             return aString;
          }

          /*****************************************************************************
           *
           *              INTERNAL CLASS
           *
           *****************************************************************************/
          //This is the only interal class used in the SDT.  It can be set up as
          //an outside class but then it becomes very difficult to update the
          //scenario tree after a successful DND operation.
          //This class is the drag source listener that is used at the beginning and
          //again at the end of a DND operation.

          //To summarize how a DND operation works for the SDT:
          //FIRST: the dragGestureListener() in the libraryTree recognizes a drag
          //has begun and fetches the file
          //SECOND: the file is passed to a new ScenarioElementSetTransferable()
          //the transferable is basically responsible to identify all data types
          //(called flavors in DND speak) that are leagally drag and dropable for the
          //SDT application.  The transferable also becomes the datastructure that
          //carries the file throughout the DND operation and is the only way for
          //a drag to talk to a drop
          //THIRD: the actual java drag event is started and a new
          //ScenarioElementSetDragSourceListener() this object is created
          //FOURTH: when the mouse pointer comes into the scenario tree area a
          //dropTarget listener sees it and begins the drop portion of the operation
          //when the mouse is released the type of DND action is checked - only
          //copy is allowed all other actions are rejected.
          //the transferable is used now to get the file
          //based on the scenario element set type the name of the set is added
          //to the scenario manager
          //FIFTH: we come back here to ScenarioElementSetDragSourceListener and check
          //for drop success.  If the drop worked we update the scenario tree
```

```
public class ScenarioElementSetDragSourceListener extends DragSourceAdapter
{
   /** Creates a new instance of ScenarioElementSetDragSourceListener */
   public ScenarioElementSetDragSourceListener()
   {
   }

   public void dragDropEnd(DragSourceDropEvent event)
   {
      if(event.getDropSuccess())
      {
         int action = event.getDropAction();
         if(action == DnDConstants.ACTION_COPY)
         {
            initScenarioTree();
         }
         if(action == DnDConstants.ACTION_LINK)
         {
            JOptionPane.showMessageDialog(null, "Action Not Allowed", "Drop Event",
JOptionPane.INFORMATION_MESSAGE);
         }
         if(action == DnDConstants.ACTION_MOVE)
         {
            JOptionPane.showMessageDialog(null, "Action Not Allowed", "Drop Event",
JOptionPane.INFORMATION_MESSAGE);
         }
      }
   }
}

// Variables declaration - do not modify
private javax.swing.JMenuItem assetGoalMenuItem;
private javax.swing.JMenuItem assetMenuItem;
private javax.swing.JMenuItem buildMenuItem;
private javax.swing.JMenuItem catalogcompMenuItem;
private javax.swing.JMenuItem conditionMenuItem;
private javax.swing.JMenuItem dacGroupMenuItem;
private javax.swing.JMenuItem departmentMenuItem;
private javax.swing.JMenuItem exitMenuItem;
private javax.swing.JTextArea feedbackTextArea;
private javax.swing.JScrollPane feedbackareaScrollPane;
private javax.swing.JMenu fileMenu;
private javax.swing.JMenuItem filterMenuItem;
private javax.swing.JMenuItem integrityMenuItem;
private javax.swing.JPanel jPanel1;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JScrollPane libraryScrollPane;
private javax.swing.JTree libraryTree;
private javax.swing.JPopupMenu libraryTreePopupMenu;
private javax.swing.JMenuItem ltDeleteMenuItem;
private javax.swing.JMenuItem ltOpenMenuItem;
private javax.swing.JMenuItem ltaddMenuItem;
private javax.swing.JMenuItem networkConnectionMenuItem;
private javax.swing.JMenuItem networkMenuItem;
private javax.swing.JMenu newsubMenu;
private javax.swing.JMenuItem openMenuItem;
private javax.swing.JMenu otherMenu;
private javax.swing.JMenuItem physcompMenuItem;
private javax.swing.JMenuItem proceduralSettingsMenuItem;
private javax.swing.JMenuItem saveMenuItem;
private javax.swing.JMenuItem saveallMenuItem;
```

```
    private javax.swing.JMenuItem saveasMenuItem;
    private javax.swing.JMenuItem savescenarioMenuItem;
    private javax.swing.JMenuItem savescenarioasMenuItem;
    private javax.swing.JScrollPane scenarioScrollPane;
    private javax.swing.JTree scenarioTree;
    private javax.swing.JPopupMenu scenarioTreePopupMenu;
    private javax.swing.JMenuBar sdtMenuBar;
    private javax.swing.JToolBar sdtToolBar;
    private javax.swing.JMenuItem secrecyMenuItem;
    private javax.swing.JButton setelementdeleteButton;
    private javax.swing.JComboBox setelementmanagementComboBox;
    private javax.swing.JLabel setelementmanagementLabel;
    private javax.swing.JMenuItem stOpenMenuItem;
    private javax.swing.JMenuItem stRemoveMenuItem;
    private javax.swing.JMenu toolsMenu;
    private javax.swing.JPanel treePanel;
    private javax.swing.JMenuItem triggerMenuItem;
    private javax.swing.JMenuItem userMenuItem;
    private javax.swing.JMenuItem waAddMenuItem;
    private javax.swing.JMenuItem waRemoveMenuItem;
    private javax.swing.JPopupMenu workAreaTabPopupMenu;
    private javax.swing.JTabbedPane workareaTabbedPane;
    private javax.swing.JMenuItem workspaceMenuItem;
    private javax.swing.JMenuItem zoneMenuItem;
    // End of variables declaration

    private Scenario startupScenario;
    private java.util.Vector tabManager;
    private java.io.ObjectOutputStream output;
    private java.io.ObjectInputStream input;
    private java.io.PrintWriter buildOutput;
    private DropTarget scenarioTreeTarget;
    private int deleteTabTarget;
    private int deleteElementTarget;
    private String rightClickTreePath;
    private String datapath;
    private java.util.Vector fileContents_filterblocking;
    private java.util.Vector fileContents_filterapps;
    private java.util.Vector fileContents_basecomp;
    private java.util.Vector fileContents_os;
    private java.util.Vector fileContents_passcomplexity;
    private java.util.Vector fileContents_passlength;
    private java.util.Vector fileContents_passchangefreq;
    private java.util.Vector fileContents_software;
    private java.util.Vector fileContents_floorplan;
    private java.util.Vector fileContents_backgroundcheck;
    private java.util.Vector fileContents_organization;
    private java.util.Vector fileContents_condition;
    private java.util.Vector fileContents_trigger;
}
```

T. THE SOURCE OF:SCENARIO ELEMENT

```
/*
 * ScenarioElement.java
 *
 * Created on December 15, 2003, 3:19 PM
 */

package CCSDT;

/**
```

```
 *
 * @author  kjohns
 *
 *This is the base class for a reusable scenario element set.  By using
 *this class we can add any common methods necessary to ALL of the reusable
 *sets.  We also ensure that all reusable sets are based on Jpanel and
 *are serializable
 */
public abstract class ScenarioElement extends javax.swing.JPanel implements java.io.Serializable
{

   private String Comment;

   /** Creates a new instance of ScenarioElement */
   public ScenarioElement()
   {
   }

   public abstract String toString(String aText);

   public abstract void Save();

   public abstract void Load();

}
```

U. THE SOURCE OF: SCENARIO ELEMENT SET

```
/*
 * ScenarioElementSet.java
 *
 * Created on December 15, 2003, 3:49 PM
 */

package CCSDT;

/**
 *
 * @author  kjohns
 *
 *This class provides a way to create sets of descriptor forms.
 */
public class ScenarioElementSet implements java.io.Serializable
{

   private String elementType;
   private int lastElementUsed;
   public java.util.Vector elementContainer;
   private String containerContents[];//names of items in container
   private String elementSetName;

   /** Creates a new instance of ScenarioElementSet */
   public ScenarioElementSet()
   {
      lastElementUsed = 0;
      elementContainer = new java.util.Vector();
   }

   public void setelementType(String aType)
   {
      elementType = aType;
   }
```

```
    public String getelementType()
    {
       return elementType;
    }

    public void setlastElementUsed(int aLast)
    {
       lastElementUsed = aLast;
    }

    public int getlastElementUsed()
    {
       return lastElementUsed;
    }

    public java.util.Vector getelementContainer()
    {
       return elementContainer;
    }

    public void setelementSetName(String aName)
    {
       elementSetName = aName;
    }

    public String getelementSetName()
    {
       return elementSetName;
    }

    public int getSize()
    {
       return elementContainer.size();
    }

    public void updateContents(String aName)
    {
       containerContents[elementContainer.size()] = aName;
    }

}
```

V. THE SOURCE OF: SCENARIO ELEMENT SET DROP TARGET LISTENER

```
/*
 * ScenarioElementSetDropTargetListener.java
 *
 * Created on December 24, 2003, 11:07 AM
 */

package CCSDT;

import java.awt.*;
import java.awt.datatransfer.*;
import java.awt.event.*;
import java.awt.dnd.*;
import java.io.*;
import java.util.*;
import javax.swing.*;
import javax.swing.tree.*;
```

```
import javax.swing.JOptionPane;

/**
 *
 * @author  KJohns
 */

//To summarize how a DND operation works for the SDT:
   //FIRST: the dragGestureListener() in the libraryTree recognizes a drag
   //has begun and fetches the file
   //SECOND: the file is passed to a new ScenarioElementSetTransferable()
   //the transferable is basically responsible to identify all data types
   //(called flavors in DND speak) that are leagally drag and dropable for the
   //SDT application.  The transferable also becomes the datastructure that
   //carries the file throughout the DND operation and is the only way for
   //a drag to talk to a drop
   //THIRD: the actual java drag event is started and a new
   //ScenarioElementSetDragSourceListener() this object is created
   //FOURTH: when the mouse pointer comes into the scenario tree area a
   //dropTarget listener sees it and begins the drop portion of the operation
   //when the mouse is released the type of DND action is checked - only
   //copy is allowed all other actions are rejected.
   //the transferable is used now to get the file
   //based on the scenario element set type the name of the set is added
   //to the scenario manager
   //FIFTH: we go back to ScenarioElementSetDragSourceListener and check
   //for drop success.  If the drop worked we update the scenario tree
public class ScenarioElementSetDropTargetListener implements DropTargetListener
{

      /** Creates a new instance of ScenarioElementSetDropTargetListener */
      public ScenarioElementSetDropTargetListener(Scenario aScenario, JTree aTree)
      {
         localScenario = aScenario;
         localTree = aTree;
      }

      public boolean isDragAcceptable(DropTargetDragEvent event)
      {
         //a drag is acceptable if the action types is correct
         //the program will only accept copy no move
         return (event.getDropAction() & DnDConstants.ACTION_COPY) != 0;
      }

      public boolean isDropAcceptable(DropTargetDropEvent event)
      {
         //same test as isDragAcceptable()!!
         //a drop is acceptable if the action types is correct
         //the program will only accept copy no move
         return (event.getDropAction() & DnDConstants.ACTION_COPY) != 0;
      }

      public void dragEnter(DropTargetDragEvent event)
      {
         if(!isDragAcceptable(event))
         {
            event.rejectDrag();
            return;
         }
      }

      public void dragExit(DropTargetEvent event)
```

```
{
}

public void dragOver(DropTargetDragEvent event)
{
   //visual feedback goes here e.g. mouse pointer changes to a custom
   //icon - o.w. the default icons are used.
}

public void dropActionChanged(DropTargetDragEvent event)
{
   //allows the user to change his/her mind and try to do a
   //move or link
   //In this implementation such a change is not possible
   //in other words, all drop actions but copy are
   //rejected
   if(!isDragAcceptable(event))
   {
      event.rejectDrag();
      return;
   }
}

public void drop(DropTargetDropEvent event)
{
   //deals with the release of the mouse button
   if(!isDropAcceptable(event))
   {
      event.rejectDrop();
      return;
   }
   event.acceptDrop(DnDConstants.ACTION_COPY);
   //the transferable holds a ScenarioElementSet
   Transferable transferable = event.getTransferable();
   Object tempObject;
   Object tempVectorObject;
   java.util.Vector tempVector;
   ScenarioElementSet tempSet;
   try
   {
      tempObject = transferable.getTransferData(networkDataFlavor);
      tempSet = (ScenarioElementSet)tempObject;
      String assetStr = "Asset";
      String assetGoalStr = "Asset Goal";
      String cataStr = "Catalog Component";
      String conditionStr = "Condition";
      String dacStr = "DAC Group";
      String deptStr = "Department";
      String filterStr = "Filter";
      String intStr = "Integrity";
      String netStr = "Network";
      String physStr = "Physical Component";
      String procStr = "Procedural Settings";
      String compnetStr = "Network Connection";
      String secStr = "Secrecy";
      String triggerStr = "Trigger";
      String userStr = "User";
      String workspaceStr = "Workspace";
      String zoneStr = "Zone";
      if(tempSet.getelementType().equalsIgnoreCase(assetStr))
      {
         tempVectorObject = localScenario.scenarioManager.get(0);
```

```
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(assetGoalStr))
{
   tempVectorObject = localScenario.scenarioManager.get(1);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(cataStr))
{
   tempVectorObject = localScenario.scenarioManager.get(2);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(conditionStr))
{
   tempVectorObject = localScenario.scenarioManager.get(3);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(dacStr))
{
   tempVectorObject = localScenario.scenarioManager.get(4);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(deptStr))
{
   tempVectorObject = localScenario.scenarioManager.get(5);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(intStr))
{
   tempVectorObject = localScenario.scenarioManager.get(6);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(netStr))
{
   tempVectorObject = localScenario.scenarioManager.get(7);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(physStr))
{
   tempVectorObject = localScenario.scenarioManager.get(8);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(secStr))
{
   tempVectorObject = localScenario.scenarioManager.get(9);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
}
else if(tempSet.getelementType().equalsIgnoreCase(triggerStr))
{
   tempVectorObject = localScenario.scenarioManager.get(10);
   tempVector = (java.util.Vector)tempVectorObject;
   tempVector.addElement(tempSet.getelementSetName());
```

```
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(userStr))
                    {
                       tempVectorObject = localScenario.scenarioManager.get(11);
                       tempVector = (java.util.Vector)tempVectorObject;
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(workspaceStr))
                    {
                       tempVectorObject = localScenario.scenarioManager.get(12);
                       tempVector = (java.util.Vector)tempVectorObject;
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(zoneStr))
                    {
            //               System.out.println("In drop() FOUND ZONE DATA FLAVOR...\n");
                       tempVectorObject = localScenario.scenarioManager.get(13);
                       tempVector = (java.util.Vector)tempVectorObject;
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(filterStr))
                    {
            //             System.out.println("In drop() FOUND FILTER DATA FLAVOR...\n");
                       tempVectorObject = localScenario.scenarioManager.get(14);
                       tempVector = (java.util.Vector)tempVectorObject;//the other vector - still need to get the filter
vector - index 0
                       tempVectorObject = tempVector.get(0);
                       tempVector = (java.util.Vector)tempVectorObject;//the filter
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(procStr))
                    {
                       tempVectorObject = localScenario.scenarioManager.get(14);
                       tempVector = (java.util.Vector)tempVectorObject;//the other vector - still need to get the
procedural settings vector - index 1
                       tempVectorObject = tempVector.get(1);
                       tempVector = (java.util.Vector)tempVectorObject;//the procedural settings
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                    else if(tempSet.getelementType().equalsIgnoreCase(compnetStr))
                    {
                       tempVectorObject = localScenario.scenarioManager.get(14);
                       tempVector = (java.util.Vector)tempVectorObject;//the other vector - still need to get the
network connection settings vector - index 1
                       tempVectorObject = tempVector.get(2);
                       tempVector = (java.util.Vector)tempVectorObject;//the network connection
                       tempVector.addElement(tempSet.getelementSetName());
                    }
                 }
                 catch(Exception e)
                 {
                    JOptionPane.showMessageDialog(null, "Exception during drop calculation.  Only reusable sets can
be dropped directories and any other data type are not allowed.", "Exception", JOptionPane.ERROR_MESSAGE);
                 }

                 event.dropComplete(true);
              }

              private Scenario localScenario;
              private JTree localTree;
              private ObjectNode localObjectNode;
              private java.util.List flavorList;
```

```
            private DataFlavor assetDataFlavor = new DataFlavor(new Asset().getClass(),"Asset");
            private DataFlavor assetGoalDataFlavor = new DataFlavor(new AssetGoal().getClass(),"Asset Goal");
            private DataFlavor catalogcomponentDataFlavor = new DataFlavor(new
CatalogComponent().getClass(),"Catalog Component");
            private DataFlavor conditionDataFlavor = new DataFlavor(new Condition().getClass(),"Condition");
            private DataFlavor dacGroupDataFlavor = new DataFlavor(new DACGroup().getClass(),"DAC
Group");
            private DataFlavor departmentDataFlavor = new DataFlavor(new
Department().getClass(),"Department");
            private DataFlavor filterDataFlavor = new DataFlavor(new Filter().getClass(),"Filter");
            private DataFlavor integrityDataFlavor = new DataFlavor(new Integrity().getClass(),"Integrity");
            private DataFlavor networkDataFlavor = new DataFlavor(new Network().getClass(),"Network");
            private DataFlavor physicalcomponentDataFlavor = new DataFlavor(new
PhysicalComponent().getClass(),"Physical Component");
            private DataFlavor proceduralSettingsDataFlavor = new DataFlavor(new
ProceduralSettings().getClass(),"Procedural Settings");
            private DataFlavor NetworkConnectionDataFlavor = new DataFlavor(new
NetworkConnection().getClass(),"Procedural Settings");
            private DataFlavor secrecyDataFlavor = new DataFlavor(new Secrecy().getClass(),"Secrecy");
            private DataFlavor triggerDataFlavor = new DataFlavor(new Trigger().getClass(),"Trigger");
            private DataFlavor userDataFlavor = new DataFlavor(new User().getClass(),"User");
            private DataFlavor workspaceDataFlavor = new DataFlavor(new Workspace().getClass(),"Workspace");
            private DataFlavor zoneDataFlavor = new DataFlavor(new Zone().getClass(),"Zone");
        }
```

W. THE SOURCE OF: SCENARIO ELEMENT SET TRANSFERABLE

```
/*
 * ScenarioElementSetTransferable.java
 *
 * Created on December 24, 2003, 11:21 AM
 */

package CCSDT;

import java.awt.*;
import java.awt.datatransfer.*;
import java.io.*;
import java.util.*;

/**
 *
 * @author  KJohns
 *
 *This class is part of the drag and drop components
 */
   //To summarize how a DND operation works for the SDT:
   //FIRST: the dragGestureListener() in the libraryTree recognizes a drag
   //has begun and fetches the file
   //SECOND: the file is passed to a new ScenarioElementSetTransferable()
   //the transferable is basically responsible to identify all data types
   //(called flavors in DND speak) that are leagally drag and dropable for the
   //SDT application.  The transferable also becomes the datastructure that
   //carries the file throughout the DND operation and is the only way for
   //a drag to talk to a drop
   //THIRD: the actual java drag event is started and a new
   //ScenarioElementSetDragSourceListener() this object is created
   //FOURTH: when the mouse pointer comes into the scenario tree area a
   //dropTarget listener sees it and begins the drop portion of the operation
   //when the mouse is released the type of DND action is checked - only
   //copy is allowed all other actions are rejected.
   //the transferable is used now to get the file
```

```
    //based on the scenario element set type the name of the set is added
    //to the scenario manager
    //FIFTH: we go back to ScenarioElementSetDragSourceListener and check
    //for drop success.  If the drop worked we update the scenario tree

public class ScenarioElementSetTransferable implements Transferable
{

    /** Creates a new instance of ScenarioElementSetTransferable */
    public ScenarioElementSetTransferable(ScenarioElementSet aSES)
    {
        localSES = aSES;
        sesList = new ArrayList(Arrays.asList(localSES.elementContainer.toArray()));
    }

    //this method is used to get the file stored in the transferable
    public Object getTransferData(DataFlavor flavor) throws UnsupportedFlavorException, IOException
    {
        if(flavor.equals(assetDataFlavor) ||
            flavor.equals(assetGoalDataFlavor) ||
            flavor.equals(catalogcomponentDataFlavor) ||
            flavor.equals(conditionDataFlavor) ||
            flavor.equals(dacGroupDataFlavor) ||
            flavor.equals(departmentDataFlavor) ||
            flavor.equals(filterDataFlavor) ||
            flavor.equals(integrityDataFlavor) ||
            flavor.equals(networkDataFlavor) ||
            flavor.equals(physicalcomponentDataFlavor) ||
            flavor.equals(proceduralSettingsDataFlavor) ||
            flavor.equals(NetworkConnectionDataFlavor) ||
            flavor.equals(secrecyDataFlavor) ||
            flavor.equals(triggerDataFlavor) ||
            flavor.equals(userDataFlavor) ||
            flavor.equals(workspaceDataFlavor) ||
            flavor.equals(zoneDataFlavor))
        {
            Object tempObject = localSES;
            return tempObject;
        }
        else
            throw new UnsupportedFlavorException(flavor);
    }

    public DataFlavor[] getTransferDataFlavors()
    {
        return flavors;
    }

    public boolean isDataFlavorSupported(DataFlavor flavor)
    {
        return Arrays.asList(flavors).contains(flavor);
    }

    private ScenarioElementSet localSES;
    private java.util.List sesList;
    private DataFlavor assetDataFlavor = new DataFlavor(new Asset().getClass(),"Asset");
    private DataFlavor assetGoalDataFlavor = new DataFlavor(new AssetGoal().getClass(),"Asset Goal");
    private DataFlavor catalogcomponentDataFlavor = new DataFlavor(new
CatalogComponent().getClass(),"Catalog Component");
    private DataFlavor conditionDataFlavor = new DataFlavor(new Condition().getClass(),"Condition");
    private DataFlavor dacGroupDataFlavor = new DataFlavor(new DACGroup().getClass(),"DAC Group");
    private DataFlavor departmentDataFlavor = new DataFlavor(new Department().getClass(),"Department");
```

```
            private DataFlavor filterDataFlavor = new DataFlavor(new Filter().getClass(),"Filter");
            private DataFlavor integrityDataFlavor = new DataFlavor(new Integrity().getClass(),"Integrity");
            private DataFlavor networkDataFlavor = new DataFlavor(new Network().getClass(),"Network");
            private DataFlavor physicalcomponentDataFlavor = new DataFlavor(new
PhysicalComponent().getClass(),"Physical Component");
            private DataFlavor proceduralSettingsDataFlavor = new DataFlavor(new
ProceduralSettings().getClass(),"Procedural Settings");
            private DataFlavor NetworkConnectionDataFlavor = new DataFlavor(new
NetworkConnection().getClass(),"Procedural Settings");
            private DataFlavor secrecyDataFlavor = new DataFlavor(new Secrecy().getClass(),"Secrecy");
            private DataFlavor triggerDataFlavor = new DataFlavor(new Trigger().getClass(),"Trigger");
            private DataFlavor userDataFlavor = new DataFlavor(new User().getClass(),"User");
            private DataFlavor workspaceDataFlavor = new DataFlavor(new Workspace().getClass(),"Workspace");
            private DataFlavor zoneDataFlavor = new DataFlavor(new Zone().getClass(),"Zone");
            private DataFlavor[] flavors
={assetDataFlavor,assetGoalDataFlavor,catalogcomponentDataFlavor,conditionDataFlavor,dacGroupDataFlavor,depar
tmentDataFlavor,filterDataFlavor,integrityDataFlavor,networkDataFlavor,physicalcomponentDataFlavor,proceduralSet
tingsDataFlavor,NetworkConnectionDataFlavor,secrecyDataFlavor,triggerDataFlavor,userDataFlavor,workspaceDataF
lavor,zoneDataFlavor};

        }
```

X. THE SOURCE OF: SECRECY

```
        /*
         * Secrecy.java
         *
         * Created on December 16, 2003, 9:20 AM
         */

        package CCSDT;

        /**
         *
         * @author  kjohns
         *
         *This class is the graphical representation of a Secrecy descriptor.
         *This is where the Secrecy form is managed and the Secrecy
         *toString() resides.
         */
        public class Secrecy extends ScenarioElement implements java.io.Serializable
        {

            /** Creates new form Secrecy */
            public Secrecy()
            {
                initComponents();
            }

            private void initComponents() {
                java.awt.GridBagConstraints gridBagConstraints;

                secrecyScrollPane = new javax.swing.JScrollPane();
                secrecyPanel = new javax.swing.JPanel();
                nameLabel = new javax.swing.JLabel();
                nameTextField = new javax.swing.JTextField();
                levelLabel = new javax.swing.JLabel();
                levelTextField = new javax.swing.JTextField();
                categoryLabel = new javax.swing.JLabel();
                valueLabel = new javax.swing.JLabel();
                valueTextField = new javax.swing.JTextField();
                valuechangeLabel = new javax.swing.JLabel();
```

```
valuechangeTextField = new javax.swing.JTextField();
attackervalueLabel = new javax.swing.JLabel();
attackervalueTextField = new javax.swing.JTextField();
attackervaluechangeLabel = new javax.swing.JLabel();
attackervaluechangeTextField = new javax.swing.JTextField();
initialbackgroundcheckLabel = new javax.swing.JLabel();
initialbackgroundcheckComboBox = new javax.swing.JComboBox();
categoryTextField = new javax.swing.JTextField();

setLayout(new java.awt.GridLayout(1, 0));

secrecyPanel.setLayout(new java.awt.GridBagLayout());

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Secrecy Tag Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(nameTextField, gridBagConstraints);

levelLabel.setLabelFor(levelTextField);
levelLabel.setText("Secrecy Level:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(levelLabel, gridBagConstraints);

levelTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(levelTextField, gridBagConstraints);

categoryLabel.setText("Secrecy Category:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(categoryLabel, gridBagConstraints);

valueLabel.setLabelFor(valueTextField);
valueLabel.setText("Value of Secret:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(valueLabel, gridBagConstraints);

valueTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(valueTextField, gridBagConstraints);

valuechangeLabel.setLabelFor(valuechangeTextField);
```

```
valuechangeLabel.setText("Monthly Change in Secrets Value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(valuechangeLabel, gridBagConstraints);

valuechangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(valuechangeTextField, gridBagConstraints);

attackervalueLabel.setLabelFor(attackervalueTextField);
attackervalueLabel.setText("Value of Secret for an attacker:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(attackervalueLabel, gridBagConstraints);

attackervalueTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(attackervalueTextField, gridBagConstraints);

attackervaluechangeLabel.setLabelFor(attackervaluechangeTextField);
attackervaluechangeLabel.setText("Monthly Change in Attacker Value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(attackervaluechangeLabel, gridBagConstraints);

attackervaluechangeTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(attackervaluechangeTextField, gridBagConstraints);

initialbackgroundcheckLabel.setLabelFor(initialbackgroundcheckComboBox);
initialbackgroundcheckLabel.setText("Initial Background Check:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
secrecyPanel.add(initialbackgroundcheckLabel, gridBagConstraints);

initialbackgroundcheckComboBox.setPreferredSize(new java.awt.Dimension(150, 25));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
secrecyPanel.add(initialbackgroundcheckComboBox, gridBagConstraints);

categoryTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
```

```
    gridBagConstraints.gridy = 2;
    gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
    secrecyPanel.add(categoryTextField, gridBagConstraints);

    secrecyScrollPane.setViewportView(secrecyPanel);

    add(secrecyScrollPane);

}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any  lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
      // if the target of the vector is a list box just add the
      // vector directly
public void populateStaticSourceLists(java.util.Vector aBackgroundcheck)
{
    //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
    //to populate:
    //background check

    //base component
    for(int i = 0; i < aBackgroundcheck.size(); i++)
    {
        initialbackgroundcheckComboBox.addItem(aBackgroundcheck.get(i));
    }
}

public void Load()
{
}

public void Save()
{
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an secrecy descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString +=
    "Secrecy:\n"+
    "\n\t//A text string to use as a secrecy label tag in this scenario"+
    " file\n\t//for example, \"Top Secret\"\n"+
    "\tName: "+nameTextField.getText()+" :end\n\n"+
    "\t//A single value representing the secrecy value between 0 and 64\n"+
    "\tLevel: "+levelTextField.getText()+
```

```
    " :end \t//an enumerated value--0 = none\n\n"+
    "\t//an enumeration of secrecy categories, e.g., 1,4,7.  Alternately"+
    " the\n\t//value can be \"none\" \n"+
    "\tCategory: "+categoryTextField.getText()+
    " :end\n\n"+
    "\t//Secrecy value is how valuable it is for the organization to keep"+
    " this a secret\n"+
    "\tSecrecyValue: "+valueTextField.getText()+" :end\n\n"+
    "\t//Integer factor from -10 to +10 determining how much the value"+
    " changes per\n\t//month.  -10 is max drop per month, 0 is no change"+
    " per month, +10 is max\n\t//increase per month\n"+
    "\tSecrecyValueChange: "+valuechangeTextField.getText()+" :end\n\n"+
    "\t//Attacker Value is what this is worth to an attacker to break its"+
    " secrecy\n"+
    "\tAttackerValue: "+attackervalueTextField.getText()+" :end\n\n"+
    "\t//Integer factor from -10 to +10 determining how much the value"+
    " changes per\n\t//month.  -10 is max drop per month, 0 is no change"+
    " per month, +10 is max\n\t//increase per month\n"+
    "\tAttackerValueChange: "+attackervaluechangeTextField.getText()+
    " :end\n\n"+
    "\t//This is what type of check someone needs\n"+
    "\tInitialBackGroundCheck: "+
    initialbackgroundcheckComboBox.getSelectedItem().toString()+
    " :end\t//High,Med,Low,None\n\n"+
    ":end //of Secrecy\n\n";
    return outputString;
}

public String getNameTextField()
{
    return nameTextField.getText();
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(java.util.Vector aBackgroundcheck)
{
    populateStaticSourceLists(aBackgroundcheck);
}

// Variables declaration - do not modify
private javax.swing.JLabel attackervalueLabel;
private javax.swing.JTextField attackervalueTextField;
private javax.swing.JLabel attackervaluechangeLabel;
private javax.swing.JTextField attackervaluechangeTextField;
private javax.swing.JLabel categoryLabel;
private javax.swing.JTextField categoryTextField;
private javax.swing.JComboBox initialbackgroundcheckComboBox;
private javax.swing.JLabel initialbackgroundcheckLabel;
private javax.swing.JLabel levelLabel;
private javax.swing.JTextField levelTextField;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
private javax.swing.JPanel secrecyPanel;
private javax.swing.JScrollPane secrecyScrollPane;
private javax.swing.JLabel valueLabel;
private javax.swing.JTextField valueTextField;
private javax.swing.JLabel valuechangeLabel;
private javax.swing.JTextField valuechangeTextField;
// End of variables declaration
```

```
}
```

Y. THE SOURCE OF: TRIGGER

```
/*
 * Trigger.java
 *
 * Created on January 23, 2004, 10:05 AM
 */

package CCSDT;
import javax.swing.JOptionPane;
import java.io.*;
/**
 *
 * @author  KJohns
 *
 *This class is the graphical representation of a Trigger descriptor.
 *This is where the Trigger form is managed and the Trigger
 *toString() resides.
 */
public class Trigger extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Trigger */
    public Trigger()
    {
        initComponents();
        triggerINIVec = new java.util.Vector();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        triggerScrollPane = new javax.swing.JScrollPane();
        triggerPanel = new javax.swing.JPanel();
        triggerInfoPanel = new javax.swing.JPanel();
        nameTextField = new javax.swing.JTextField();
        nameLabel = new javax.swing.JLabel();
        triggerClassLabel = new javax.swing.JLabel();
        triggerClassComboBox = new javax.swing.JComboBox();
        triggerClassTextField = new javax.swing.JTextField();
        frequencyLabel = new javax.swing.JLabel();
        frequencyTextField = new javax.swing.JTextField();
        fixedDelayTextField = new javax.swing.JTextField();
        fixedDelayLabel = new javax.swing.JLabel();
        randomDelayLabel = new javax.swing.JLabel();
        randomDelayTextField = new javax.swing.JTextField();
        paramListPanel = new javax.swing.JPanel();
        parameter1TextField = new javax.swing.JTextField();
        parameter2TextField = new javax.swing.JTextField();
        parameter3TextField = new javax.swing.JTextField();
        parameter4TextField = new javax.swing.JTextField();
        parameter5TextField = new javax.swing.JTextField();
        parameter5Label = new javax.swing.JLabel();
        parameter4Label = new javax.swing.JLabel();
        parameter3Label = new javax.swing.JLabel();
        parameter2Label = new javax.swing.JLabel();
        parameter1Label = new javax.swing.JLabel();
        conditionListPanel = new javax.swing.JPanel();
        convenienceListingPanel = new javax.swing.JPanel();
        conditionListScrollPane = new javax.swing.JScrollPane();
```

```
conditionListTextArea = new javax.swing.JTextArea();
conditionListTextField = new javax.swing.JTextField();
firingConditionLabel = new javax.swing.JLabel();

setLayout(new java.awt.GridLayout(1, 0));

triggerPanel.setLayout(new java.awt.GridBagLayout());

triggerPanel.setPreferredSize(new java.awt.Dimension(1035, 516));
triggerInfoPanel.setLayout(new java.awt.GridBagLayout());

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(nameTextField, gridBagConstraints);

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Trigger Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
triggerInfoPanel.add(nameLabel, gridBagConstraints);

triggerClassLabel.setText("Trigger Class:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
triggerInfoPanel.add(triggerClassLabel, gridBagConstraints);

triggerClassComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
triggerClassComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        triggerClassComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(triggerClassComboBox, gridBagConstraints);

triggerClassTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(triggerClassTextField, gridBagConstraints);

frequencyLabel.setText("Frequency:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
triggerInfoPanel.add(frequencyLabel, gridBagConstraints);

frequencyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(frequencyTextField, gridBagConstraints);

fixedDelayTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(fixedDelayTextField, gridBagConstraints);

fixedDelayLabel.setText("Fixed Delay:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
triggerInfoPanel.add(fixedDelayLabel, gridBagConstraints);

randomDelayLabel.setText("Random Delay:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
triggerInfoPanel.add(randomDelayLabel, gridBagConstraints);

randomDelayTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(randomDelayTextField, gridBagConstraints);

paramListPanel.setLayout(new java.awt.GridBagLayout());

paramListPanel.setBorder(new javax.swing.border.TitledBorder("Parameter List"));
parameter1TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
paramListPanel.add(parameter1TextField, gridBagConstraints);

parameter2TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
paramListPanel.add(parameter2TextField, gridBagConstraints);

parameter3TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
paramListPanel.add(parameter3TextField, gridBagConstraints);

parameter4TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
```

```
paramListPanel.add(parameter4TextField, gridBagConstraints);

parameter5TextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
paramListPanel.add(parameter5TextField, gridBagConstraints);

parameter5Label.setText("Parameter 5:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
paramListPanel.add(parameter5Label, gridBagConstraints);

parameter4Label.setText("Parameter 4:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
paramListPanel.add(parameter4Label, gridBagConstraints);

parameter3Label.setText("Parameter 3:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
paramListPanel.add(parameter3Label, gridBagConstraints);

parameter2Label.setText("Parameter 2:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
paramListPanel.add(parameter2Label, gridBagConstraints);

parameter1Label.setText("Parameter 1:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
paramListPanel.add(parameter1Label, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
triggerInfoPanel.add(paramListPanel, gridBagConstraints);

triggerPanel.add(triggerInfoPanel, new java.awt.GridBagConstraints());

conditionListPanel.setLayout(new java.awt.GridBagLayout());

convenienceListingPanel.setLayout(new java.awt.GridBagLayout());

convenienceListingPanel.setBorder(new javax.swing.border.TitledBorder("All Available Conditions
(Convenience Listing)"));
conditionListScrollPane.setPreferredSize(new java.awt.Dimension(1000, 150));
conditionListTextArea.setEditable(false);
```

```
        conditionListScrollPane.setViewportView(conditionListTextArea);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(0, 0, 25, 0);
        conveniencelistingPanel.add(conditionListScrollPane, gridBagConstraints);

        conditionListPanel.add(conveniencelistingPanel, new java.awt.GridBagConstraints());

        conditionListTextField.setPreferredSize(new java.awt.Dimension(700, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 3;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        conditionListPanel.add(conditionListTextField, gridBagConstraints);

        firingConditionLabel.setText("Trigger Firing Condition (Boolean Expression):");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        conditionListPanel.add(firingConditionLabel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 10;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        triggerPanel.add(conditionListPanel, gridBagConstraints);

        triggerScrollPane.setViewportView(triggerPanel);

        add(triggerScrollPane);

    }
//the following listener sets labels based on the type of trigger
    private void triggerClassComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(triggerClassComboBox.getSelectedItem() != null)
        {
            triggerClassTextField.setText(
                        triggerClassComboBox.getSelectedItem().toString());
            Object tempObj = new Object();
            java.util.Vector tempVec = new java.util.Vector();
            tempObj = triggerINIVec.get(
                            triggerClassComboBox.getSelectedIndex());
            tempVec = (java.util.Vector)tempObj;
            for(int i = 0; i < tempVec.size(); i++)
            {
                switch(i)
                {
                    case 1:
                        parameter1Label.setText(tempVec.get(1).toString()+":");
                        parameter2Label.setText("Parameter 2:");
                        parameter3Label.setText("Parameter 3:");
                        parameter4Label.setText("Parameter 4:");
                        parameter5Label.setText("Parameter 5:");
                        break;
                    case 2:
```

```
                parameter2Label.setText(tempVec.get(2).toString()+":");
                parameter3Label.setText("Parameter 3:");
                parameter4Label.setText("Parameter 4:");
                parameter5Label.setText("Parameter 5:");
                break;
            case 3:
                parameter3Label.setText(tempVec.get(3).toString()+":");
                parameter4Label.setText("Parameter 4:");
                parameter5Label.setText("Parameter 5:");
                break;
            case 4:
                parameter4Label.setText(tempVec.get(4).toString()+":");
                parameter5Label.setText("Parameter 5:");
                break;
            case 5:
                parameter5Label.setText(tempVec.get(5).toString()+":");
                break;
            default:
                parameter1Label.setText("Parameter 1:");
                parameter2Label.setText("Parameter 2:");
                parameter3Label.setText("Parameter 3:");
                parameter4Label.setText("Parameter 4:");
                parameter5Label.setText("Parameter 5:");
                break;
        }
      }
   }
}

//Interface combo box and list box content comes from one of two sources
//the content is either from ini files or from reusable sets that have
//already been added to the scenario in development.
//If the content come from ini files it is designated static.  This is
//because the content of ini files does not change.
//If the content comes from reusable sets that have been added to the
//scenario in development it is designated dynamic.  This is because
//the content may be empty, of any lenght and change frequently

//populateStaticSourceLists() takes a list of vectors as
//parameters in ScenarioDefinitionTool.java and is used to add static
//content to combo boxes and list boxes.
//First any combo boxes are cleared and reinitialized
//Second if the target of the vector is a combo box - for each
//element in the vector add it to the combo box
     // if the target of the vector is a list box just add the
     // vector directly
public void populateStaticSourceLists(java.util.Vector aTrigger)
{
   //uses the vectors collected by readINIFile() in ScenarioDefinitionTool
   //to populate:
   //triggers

   triggerINIVec = aTrigger;
   //triggers
   triggerClassComboBox.removeAllItems();
   Object tempObj = new Object();
   java.util.Vector tempVec = new java.util.Vector();
   for(int i = 0; i < aTrigger.size(); i++)
   {
      tempObj = aTrigger.get(i);
      tempVec = (java.util.Vector)tempObj;
      triggerClassComboBox.addItem(tempVec.get(0));
```

```
    }
  }

  //populateDynamicSourceLists() takes datapath and startupScenario as
  //parameters in ScenarioDefinitionTool.java and is used to add dynamic
  //content to combo boxes and list boxes.
  //First the scenairo is used to get the added reusable sets
  //Second any combo boxes to be used are cleared and reinitialized
  //Third for each reusable set added a file input object is created
  //Fourth for each member of the set the element name is added to the
  //combo box or list.
  public void populateDynamicSourceLists(String aPath, Scenario aScenario)
  {
    //uses the Scenario passed to it to set the dynamic values of
    //dependent sets
    //Conditions
    mScenario = aScenario;
    Object tempObj = new Object();
    java.util.Vector tempVec = new java.util.Vector();
    ScenarioElementSet tempSet = new ScenarioElementSet();

    //Conditions
    conditionListTextArea.removeAll();
    tempObj = aScenario.scenarioManager.get(3);
    tempVec = (java.util.Vector)tempObj;
    for(int i = 0; i < tempVec.size(); i++)
    {
      tempObj = tempVec.get(i);
      try
      {
        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Condition/"+tempObj.toString()));
      }
      catch(IOException ioException)
      {
        JOptionPane.showMessageDialog(this,
            "Exception during input stream creation", "Exception",
            JOptionPane.ERROR_MESSAGE);
      }
      try
      {
        tempSet = (ScenarioElementSet)input.readObject();
      }
      catch(ClassNotFoundException classnotfoundException)
      {
        JOptionPane.showMessageDialog(this,
            "Exception during file read - the object was not found",
            "Exception", JOptionPane.ERROR_MESSAGE);
      }
      catch(IOException ioException)
      {
        JOptionPane.showMessageDialog(this, "Exception during file read",
            "Exception", JOptionPane.ERROR_MESSAGE);
      }
      try
      {
        input.close();
      }
      catch(IOException ioException)
      {
        JOptionPane.showMessageDialog(this, "Exception during file close",
            "Exception", JOptionPane.ERROR_MESSAGE);
```

```
        }
        for(int j = 0; j < tempSet.elementContainer.size(); j++)
        {
           Condition tempCondition = new Condition();
           tempObj = tempSet.elementContainer.get(j);
           tempCondition = (Condition)tempObj;
           conditionListTextArea.append(tempCondition.getTriggerString());
        }
     }
  }

  public void Load()
  {
  }

  public void Save()
  {
  }

  //Each descriptor form has a toString() method that is used by build() in
  //the SDT to create the SDF.

  //creates the formatted output for an trigger descriptor
  public String toString(String aText)
  {
     String outputString = new String();
     outputString +=
     "\tTrigger:\n\n"+
     "\t\t//Enumeration value that defines the trigger class\n\n"+
     "\t\t//For example,  WinTrigger, LoseTrigger, LogTrigger, MessageTrigger,\n"+
     "\t\t//AttackTrigger, TickerTrigger etc.\n"+
     "\t\tTriggerClass: "+triggerClassTextField.getText()+" :end\n\n"+
     "\t\t//A simple text name used for event logging and debugging\n"+
     "\t\tTriggerName: "+nameTextField.getText()+" :end\n\n"+
     "\t\t//The frequency in days of this event, e.g., “5” means once per 5 days.\n"+
     "\t\t//And 0.5 means twice a day.\n"+
     "\t\tFrequencyInDays: "+frequencyTextField.getText()+" :end\n\n"+
     "\t\t//A fixed number of days (expressed as a float) after which a triggered\n"+
     "\t\t//event will occur.\n"+
     "\t\tFixedDelay: "+fixedDelayTextField.getText()+" :end\n\n"+
     "\t\t//A random number of days (expressed as a float) after which a triggered\n"+
     "\t\t//event will occur.  The resulting delay will be a normal distribution \n"+
     "\t\t//between zero and the given value.  This value will be added to\n"+
     "\t\t//the FixedDelay value (if any).\n"+
     "\t\tRandomDelay: "+randomDelayTextField.getText()+" :end\n\n"+
     "\t\t//The trigger class determines the number of parameters and their\n"+
     "\t\t//type (i.e. integer, string, boolean, enumeration)\n\n"+
     "\t\t//The number and type of parameters is fixed for each trigger class.\n";
     if(parameter1TextField.getText().length() > 0)
     {
        outputString += "\n\t\tParameter: "+parameter1TextField.getText()+" :end";
     }
     if(parameter2TextField.getText().length() > 0)
     {
        outputString += "\n\t\tParameter: "+parameter2TextField.getText()+" :end";
     }
     if(parameter3TextField.getText().length() > 0)
     {
        outputString += "\n\t\tParameter: "+parameter3TextField.getText()+" :end";
     }
     if(parameter4TextField.getText().length() > 0)
     {
```

```
            outputString += "\n\t\tParameter: "+parameter4TextField.getText()+" :end";
        }
        if(parameter5TextField.getText().length() > 0)
        {
            outputString += "\n\t\tParameter: "+parameter5TextField.getText()+" :end";
        }
        outputString +="\n\n\t\t//What conditions trigger this trigger.\n"+
        "\t\t//AND,  OR,  NOT and ( ) are tokens\n\n "+
        "\t\t//The list is a list of condition tagnames connected by the “AND”, “OR”, \n//“NOT”, “(“, “)” t okens
terminated by the :end token\n\n"+
        "\t\t//An example list would be:\n"+
        "\t\t//(tagname2  AND NOT  tagname4) OR tagname5\n"+
        "\t\tConditionList: "+conditionListTextField.getText()+" :end\n\n"+
        "\t:end\n\n";



        return outputString;
    }

    public String getNameTextField()
    {
        return nameTextField.getText();
    }

    //This method provides a way to reinitialize any miscellaneous
    //(non-list, non-button) listeners when they die after IO operations.
    //The code is taken line for line from  the initComponents() method.
    public void reInitListeners(String aPath, Scenario aScenario, java.util.Vector aTrigger)
    {
        populateDynamicSourceLists(aPath, aScenario);
        populateStaticSourceLists(aTrigger);
        triggerClassComboBox.addActionListener(new java.awt.event.ActionListener()
        {
            public void actionPerformed(java.awt.event.ActionEvent evt)
            {
                triggerClassComboBoxActionPerformed(evt);
            }
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JPanel conditionListPanel;
    private javax.swing.JScrollPane conditionListScrollPane;
    private javax.swing.JTextArea conditionListTextArea;
    private javax.swing.JTextField conditionListTextField;
    private javax.swing.JPanel convenienceListingPanel;
    private javax.swing.JLabel firingConditionLabel;
    private javax.swing.JLabel fixedDelayLabel;
    private javax.swing.JTextField fixedDelayTextField;
    private javax.swing.JLabel frequencyLabel;
    private javax.swing.JTextField frequencyTextField;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JPanel paramListPanel;
    private javax.swing.JLabel parameter1Label;
    private javax.swing.JTextField parameter1TextField;
    private javax.swing.JLabel parameter2Label;
    private javax.swing.JTextField parameter2TextField;
    private javax.swing.JLabel parameter3Label;
    private javax.swing.JTextField parameter3TextField;
    private javax.swing.JLabel parameter4Label;
```

```
    private javax.swing.JTextField parameter4TextField;
    private javax.swing.JLabel parameter5Label;
    private javax.swing.JTextField parameter5TextField;
    private javax.swing.JLabel randomDelayLabel;
    private javax.swing.JTextField randomDelayTextField;
    private javax.swing.JComboBox triggerClassComboBox;
    private javax.swing.JLabel triggerClassLabel;
    private javax.swing.JTextField triggerClassTextField;
    private javax.swing.JPanel triggerInfoPanel;
    private javax.swing.JPanel triggerPanel;
    private javax.swing.JScrollPane triggerScrollPane;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private java.util.Vector triggerINIVec;
    private Scenario mScenario;
}
```

Z. THE SOURCE OF: USER

```
/*
 * User.java
 *
 * Created on January 23, 2004, 10:08 AM
 */

package CCSDT;
import javax.swing.*;
import javax.swing.JList;
import java.io.*;
/**
 *
 * @author  KJohns
 *
 *
 *This class is the graphical representation of a User descriptor.
 *This is where the User form is managed and the User
 *toString() resides.
 */
public class User extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form User */
    public User()
    {
        initComponents();
        dac_added = false;
        assetGoal_added = false;
        mDACGroupVec = new java.util.Vector();
        totalUserAssetGoalVec = new java.util.Vector();

        genderComboBox.addItem("Female");
        genderComboBox.addItem("Male");
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        userScrollPane = new javax.swing.JScrollPane();
        userPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        nameTextField = new javax.swing.JTextField();
        departmentLabel = new javax.swing.JLabel();
```

```
departmentComboBox = new javax.swing.JComboBox();
departmentFinalTextField = new javax.swing.JTextField();
hardwareSkillsLabel = new javax.swing.JLabel();
hardwareSkillsTextField = new javax.swing.JTextField();
softwareSkillsLabel = new javax.swing.JLabel();
softwareSkillsTextField = new javax.swing.JTextField();
peopleSkillsLabel = new javax.swing.JLabel();
peopleSkillsTextField = new javax.swing.JTextField();
availableDateLabel = new javax.swing.JLabel();
availableDateTextField = new javax.swing.JTextField();
secrecyLabel = new javax.swing.JLabel();
secrecyComboBox = new javax.swing.JComboBox();
secrecyTextField = new javax.swing.JTextField();
integrityLabel = new javax.swing.JLabel();
integrityComboBox = new javax.swing.JComboBox();
integrityTextField = new javax.swing.JTextField();
dacGroupPanel = new javax.swing.JPanel();
dacGroupLabel = new javax.swing.JLabel();
dacGroupComboBox = new javax.swing.JComboBox();
dacGroupFinalTextField = new javax.swing.JTextField();
dacAddButton = new javax.swing.JButton();
dacScrollPane = new javax.swing.JScrollPane();
dacList = new javax.swing.JList();
dacRemoveButton = new javax.swing.JButton();
assetGoalPanel = new javax.swing.JPanel();
assetGoalLabel = new javax.swing.JLabel();
targetUsageLabel = new javax.swing.JLabel();
targetUsageTextField = new javax.swing.JTextField();
agHappinessLabel = new javax.swing.JLabel();
agHappinessTextField = new javax.swing.JTextField();
agProductivityLabel = new javax.swing.JLabel();
agProductivityTextField = new javax.swing.JTextField();
assetGoalComboBox = new javax.swing.JComboBox();
assetGroupFinalTextField = new javax.swing.JTextField();
addButton = new javax.swing.JButton();
assetGoalScrollPane = new javax.swing.JScrollPane();
assetGoalList = new javax.swing.JList();
removeButton = new javax.swing.JButton();
costLabel = new javax.swing.JLabel();
costTextField = new javax.swing.JTextField();
trustworthinessLabel = new javax.swing.JLabel();
trustworthinessTextField = new javax.swing.JTextField();
posIndexLabel = new javax.swing.JLabel();
posIndexFinalComboBox = new javax.swing.JComboBox();
posIndexTextField = new javax.swing.JTextField();
initialTrainingLabel = new javax.swing.JLabel();
initialTrainingTextField = new javax.swing.JTextField();
happinessLabel = new javax.swing.JLabel();
happinessTextField = new javax.swing.JTextField();
productivityLabel = new javax.swing.JLabel();
productivityTextField = new javax.swing.JTextField();
skillLabel = new javax.swing.JLabel();
skillTextField = new javax.swing.JTextField();
genderLabel = new javax.swing.JLabel();
genderComboBox = new javax.swing.JComboBox();
descriptionScrollPane = new javax.swing.JScrollPane();
descriptionTextArea = new javax.swing.JTextArea();
descriptionLabel = new javax.swing.JLabel();
defaultDACLabel = new javax.swing.JLabel();
defaultDACComboBox = new javax.swing.JComboBox();
defaultDACTextField = new javax.swing.JTextField();
```

```
setLayout(new java.awt.GridLayout(1, 0));

userPanel.setLayout(new java.awt.GridBagLayout());

userPanel.setPreferredSize(new java.awt.Dimension(747, 1007));
nameLabel.setLabelFor(nameTextField);
nameLabel.setText("User Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(nameTextField, gridBagConstraints);

departmentLabel.setText("Department:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(departmentLabel, gridBagConstraints);

departmentComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
departmentComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      departmentComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(departmentComboBox, gridBagConstraints);

departmentFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(departmentFinalTextField, gridBagConstraints);

hardwareSkillsLabel.setText("Hardware Skills:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(hardwareSkillsLabel, gridBagConstraints);

hardwareSkillsTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(hardwareSkillsTextField, gridBagConstraints);

softwareSkillsLabel.setText("Software Skills:");
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(softwareSkillsLabel, gridBagConstraints);

softwareSkillsTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(softwareSkillsTextField, gridBagConstraints);

peopleSkillsLabel.setText("People Skills:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(peopleSkillsLabel, gridBagConstraints);

peopleSkillsTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 5;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(peopleSkillsTextField, gridBagConstraints);

availableDateLabel.setText("Available Date:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(availableDateLabel, gridBagConstraints);

availableDateTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 6;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(availableDateTextField, gridBagConstraints);

secrecyLabel.setText("Secrecy:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(secrecyLabel, gridBagConstraints);

secrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
secrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      secrecyComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(secrecyComboBox, gridBagConstraints);

secrecyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(secrecyTextField, gridBagConstraints);

integrityLabel.setText("Integrity");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(integrityLabel, gridBagConstraints);

integrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
integrityComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        integrityComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(integrityComboBox, gridBagConstraints);

integrityTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(integrityTextField, gridBagConstraints);

dacGroupPanel.setLayout(new java.awt.GridBagLayout());

dacGroupPanel.setBorder(new javax.swing.border.TitledBorder("DAC Groups"));
dacGroupLabel.setText("DAC Group");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
dacGroupPanel.add(dacGroupLabel, gridBagConstraints);

dacGroupComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
dacGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        dacGroupComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
dacGroupPanel.add(dacGroupComboBox, gridBagConstraints);

dacGroupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
dacGroupPanel.add(dacGroupFinalTextField, gridBagConstraints);

dacAddButton.setText("Add");
dacAddButton.addActionListener(new java.awt.event.ActionListener() {
```

```
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        dacAddButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
dacGroupPanel.add(dacAddButton, gridBagConstraints);

dacScrollPane.setPreferredSize(new java.awt.Dimension(200, 150));
dacScrollPane.setViewportView(dacList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
dacGroupPanel.add(dacScrollPane, gridBagConstraints);

dacRemoveButton.setText("Remove");
dacRemoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        dacRemoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
dacGroupPanel.add(dacRemoveButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 10;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
userPanel.add(dacGroupPanel, gridBagConstraints);

assetGoalPanel.setLayout(new java.awt.GridBagLayout());

assetGoalPanel.setBorder(new javax.swing.border.TitledBorder("Assets Goals"));
assetGoalLabel.setText("Asset Goal");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
assetGoalPanel.add(assetGoalLabel, gridBagConstraints);

targetUsageLabel.setText("Taret Usage");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
assetGoalPanel.add(targetUsageLabel, gridBagConstraints);

targetUsageTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
assetGoalPanel.add(targetUsageTextField, gridBagConstraints);

agHappinessLabel.setText("Happiness");
gridBagConstraints = new java.awt.GridBagConstraints();
```

```
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
assetGoalPanel.add(agHappinessLabel, gridBagConstraints);

agHappinessTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
assetGoalPanel.add(agHappinessTextField, gridBagConstraints);

agProductivityLabel.setText("Productvity");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 48, 0, 0);
assetGoalPanel.add(agProductivityLabel, gridBagConstraints);

agProductivityTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
assetGoalPanel.add(agProductivityTextField, gridBagConstraints);

assetGoalComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
assetGoalComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        assetGoalComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
assetGoalPanel.add(assetGoalComboBox, gridBagConstraints);

assetGroupFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 5;
assetGoalPanel.add(assetGroupFinalTextField, gridBagConstraints);

addButton.setText("Add");
addButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 6;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
assetGoalPanel.add(addButton, gridBagConstraints);

assetGoalScrollPane.setPreferredSize(new java.awt.Dimension(734, 150));
assetGoalScrollPane.setViewportView(assetGoalList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
```

```
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
assetGoalPanel.add(assetGoalScrollPane, gridBagConstraints);

removeButton.setText("Remove");
removeButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        removeButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
assetGoalPanel.add(removeButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
userPanel.add(assetGoalPanel, gridBagConstraints);

costLabel.setText("Cost:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 18;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(costLabel, gridBagConstraints);

costTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 18;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(costTextField, gridBagConstraints);

trustworthinessLabel.setText("Trustworthiness:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(trustworthinessLabel, gridBagConstraints);

trustworthinessTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 12;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(trustworthinessTextField, gridBagConstraints);

posIndexLabel.setText("Grid Position in Office:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(posIndexLabel, gridBagConstraints);

posIndexFinalComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
posIndexFinalComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        posIndexFinalComboBoxActionPerformed(evt);
    }
```

```
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(posIndexFinalComboBox, gridBagConstraints);

posIndexTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(posIndexTextField, gridBagConstraints);

initialTrainingLabel.setText("Initial Training:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(initialTrainingLabel, gridBagConstraints);

initialTrainingTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(initialTrainingTextField, gridBagConstraints);

happinessLabel.setText("Happiness:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(happinessLabel, gridBagConstraints);

happinessTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(happinessTextField, gridBagConstraints);

productivityLabel.setText("Productivity:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 15;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(productivityLabel, gridBagConstraints);

productivityTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 15;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(productivityTextField, gridBagConstraints);

skillLabel.setText("Skill:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
```

```
userPanel.add(skillLabel, gridBagConstraints);

skillTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(skillTextField, gridBagConstraints);

genderLabel.setText("Gender:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 19;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(genderLabel, gridBagConstraints);

genderComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 19;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(genderComboBox, gridBagConstraints);

descriptionScrollPane.setPreferredSize(new java.awt.Dimension(425, 60));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setWrapStyleWord(true);
descriptionTextArea.setMinimumSize(new java.awt.Dimension(0, 0));
descriptionTextArea.setPreferredSize(new java.awt.Dimension(425, 1000));
descriptionScrollPane.setViewportView(descriptionTextArea);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 20;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
userPanel.add(descriptionScrollPane, gridBagConstraints);

descriptionLabel.setText("Description:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 20;
gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
gridBagConstraints.insets = new java.awt.Insets(4, 0, 0, 0);
userPanel.add(descriptionLabel, gridBagConstraints);

defaultDACLabel.setText("Default DAC Group:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
userPanel.add(defaultDACLabel, gridBagConstraints);

defaultDACComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
defaultDACComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      defaultDACComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 9;
```

```
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        userPanel.add(defaultDACComboBox, gridBagConstraints);

        defaultDACTextField.setPreferredSize(new java.awt.Dimension(150, 20));
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 9;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
        userPanel.add(defaultDACTextField, gridBagConstraints);

        userScrollPane.setViewportView(userPanel);

        add(userScrollPane);

    }
//the following listener is for the button that adds
    //entries from a list box
    private void dacAddButtonActionPerformed(java.awt.event.ActionEvent evt) {
        Object tempObject = new Object();
        mDACGroupVec.addElement(dacGroupFinalText Field.getText());
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < mDACGroupVec.size(); i++)
        {
            listModel.addElement(mDACGroupVec.get(i));
        }
        dacList = null;
        dacList = new JList(listModel);
        dacList.setSelectionMode(
                        javax.swing.ListSelectionModel.SINGLE_SELECTION);
        dacScrollPane.setViewportView(dacList);
        dac_added = true;
    }
//the following listener sets a text field based on a combo box selection
    private void posIndexFinalComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(posIndexFinalComboBox.getSelectedItem() != null)
        {
            posIndexTextField.setText(
                  String.valueOf((posIndexFinalComboBox.getSelectedIndex())-1));
        }
    }
//the following listener is for the button that removes
    //entries from a list box
    private void removeButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(assetGoalList.getSelectedIndex() >= 0)
        {
            totalUserAssetGoalVec.remove(assetGoalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)assetGoalList.getModel();
            listModel.remove(assetGoalList.getSelectedIndex());
            assetGoalList = new JList(listModel);
            assetGoalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            assetGoalScrollPane.setViewportView(assetGoalList);
        }
    }
//the following listener is for the button that adds
    //entries from a list box
    private void addButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        mUAG = new UserAssetGoal();
        Object tempObject = new Object();
```

```
        UserAssetGoal tempUAG = new UserAssetGoal();
        mUAG.setAssetGoal(assetGroupFinalTextField.getText());
        mUAG.setUsage(targetUsageTextField.getText());
        mUAG.setHappiness(agHappinessTextField.getText());
        mUAG.setProductivity(agProductivityTextField.getText());
        totalUserAssetGoalVec.addElement(mUAG);
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < totalUserAssetGoalVec.size(); i++)
        {
           tempObject = totalUserAssetGoalVec.get(i);
           tempUAG = (UserAssetGoal)tempObject;
           listModel.addElement(tempUAG.getUserAssetGoalString());
        }
        assetGoalList = null;
        assetGoalList = new JList(listModel);
        assetGoalList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
        assetGoalScrollPane.setViewportView(assetGoalList);
        assetGoal_added = true;
     }
  //the following listener sets a text field based on a combo box selection
     private void assetGoalComboBoxActionPerformed(java.awt.event.ActionEvent evt)
     {
        if(assetGoalComboBox.getSelectedItem() != null)
        {
           assetGroupFinalTextField.setText(
                    assetGoalComboBox.getSelectedItem().toString());
        }
     }
  //the following listener is for the button that removes
     //entries from a list box
     private void dacRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
     {
        if(dacList.getSelectedIndex() >= 0)
        {
           mDACGroupVec.remove(dacList.getSelectedIndex());
           DefaultListModel listModel = new DefaultListModel();
           listModel = (DefaultListModel)dacList.getModel();
           listModel.remove(dacList.getSelectedIndex());
           dacList = new JList(listModel);
           dacList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
           dacScrollPane.setViewportView(dacList);
        }
     }
  //the following listener sets a text field based on a combo box selection
     private void dacGroupComboBoxActionPerformed(java.awt.event.ActionEvent evt)
     {
       if(dacGroupComboBox.getSelectedItem() != null)
        {
           dacGroupFinalTextField.setText(
                       dacGroupComboBox.getSelectedItem().toString());
        }
     }
  //the following listener sets a text field based on a combo box selection
     private void defaultDACComboBoxActionPerformed(java.awt.event.ActionEvent evt)
     {
        if(defaultDACComboBox.getSelectedItem() != null)
        {
           defaultDACTextField.setText(
                    defaultDACComboBox.getSelectedItem().toString());
        }
     }
```

```
//the following listener sets a text field based on a combo box selection
   private void integrityComboBoxActionPerformed(java.awt.event.ActionEvent evt)
   {
      if(integrityComboBox.getSelectedItem() != null)
      {
         integrityTextField.setText(
                        integrityComboBox.getSelectedItem().toString());
      }
   }
//the following listener sets a text field based on a combo box selection
   private void secrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt)
   {
      if(secrecyComboBox.getSelectedItem() != null)
      {
         secrecyTextField.setText(
                        secrecyComboBox.getSelectedItem().toString());
      }
   }
//the following listener sets a text field based on a combo box selection
   private void departmentComboBoxActionPerformed(java.awt.event.ActionEvent evt)
   {
      if(departmentComboBox.getSelectedItem() != null)
      {
         departmentFinalTextField.setText(
                     departmentComboBox.getSelectedItem().toString());
      }
   }

   //Interface combo box and list box content comes from one of two sources
   //the content is either from ini files or from reusable sets that have
   //already been added to the scenario in development.
   //If the content come from ini files it is designated static.  This is
   //because the content of ini files does not change.
   //If the content comes from reusable sets that have been added to the
   //scenario in development it is designated dynamic.  This is because
   //the content may be empty, of any lenght and change frequently

   //populateDynamicSourceLists() takes datapath and startupScenario as
   //parameters in ScenarioDefinitionTool.java and is used to add dynamic
   //content to combo boxes and list boxes.
   //First the scenairo is used to get the added reusable sets
   //Second any combo boxes to be used are cleared and reinitialized
   //Third for each reusable set added a file input object is created
   //Fourth for each member of the set the element name is added to the
   //combo box or list.
   public void populateDynamicSourceLists(String aPath, Scenario aScenario)
   {
      //uses the Scenario passed to it to set the dynamic values of
      //dependent sets
      //Department
      //Secrecy
      //Integrity
      //DACGroup
      //AssetGoal

      mScenario = aScenario;
      Object tempObj = new Object();
      java.util.Vector tempVec = new java.util.Vector();
      ScenarioElementSet tempSet = new ScenarioElementSet();

      //Department
      tempObj = aScenario.scenarioManager.get(5);
```

```
                tempVec = (java.util.Vector)tempObj;
                departmentComboBox.removeAllItems();
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Department/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   for(int j = 0; j < tempSet.elementContainer.size(); j++)
                   {
                      Department tempDept = new Department();
                      tempObj = tempSet.elementContainer.get(j);
                      tempDept = (Department)tempObj;
                      departmentComboBox.addItem(tempDept.getNameTextField());
                   }
                }
                //DACGroup
                tempObj = aScenario.scenarioManager.get(4);
                tempVec = (java.util.Vector)tempObj;
                dacGroupComboBox.removeAllItems();
                defaultDACComboBox.removeAllItems();
                for(int i = 0; i < tempVec.size(); i++)
                {
                   tempObj = tempVec.get(i);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/DAC Group/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
```

```
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during input stream creation", "Exception",
                                JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       DACGroup tempDAC = new DACGroup();
                       tempObj = tempSet.elementContainer.get(j);
                       tempDAC = (DACGroup)tempObj;
                       dacGroupComboBox.addItem(tempDAC.getNameTextField());
                       defaultDACComboBox.addItem(tempDAC.getNameTextField());
                    }
                 }
                 //Secrecy
                 tempObj = aScenario.scenarioManager.get(9);
                 tempVec = (java.util.Vector)tempObj;
                 secrecyComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during input stream creation", "Exception",
                                JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
```

```
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file close",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                for(int j = 0; j < tempSet.elementContainer.size(); j++)
                {
                    Secrecy tempSecrecy = new Secrecy();
                    tempObj = tempSet.elementContainer.get(j);
                    tempSecrecy = (Secrecy)tempObj;
                    secrecyComboBox.addItem(tempSecrecy.getNameTextField());
                }
            }
            //Integrity
            tempObj = aScenario.scenarioManager.get(6);
            tempVec = (java.util.Vector)tempObj;
            integrityComboBox.removeAllItems();
            for(int i = 0; i < tempVec.size(); i++)
            {
                tempObj = tempVec.get(i);
                try
                {
                    input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempObj.toString()));
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    tempSet = (ScenarioElementSet)input.readObject();
                }
                catch(ClassNotFoundException classnotfoundException)
                {
                    JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                }
                catch(IOException ioException)
                {
                    JOptionPane.showMessageDialog(this, "Exception during file read",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                }
                try
                {
                    input.close();
```

```
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       Integrity tempIntegrity = new Integrity();
                       tempObj = tempSet.elementContainer.get(j);
                       tempIntegrity = (Integrity)tempObj;
                       integrityComboBox.addItem(tempIntegrity.getNameTextField());
                    }
                 }
                 //AssetGoal
                 tempObj = aScenario.scenarioManager.get(1);
                 tempVec = (java.util.Vector)tempObj;
                 assetGoalComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Goal/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       AssetGoal tempAssetGoal = new AssetGoal();
                       tempObj = tempSet.elementContainer.get(j);
                       tempAssetGoal = (AssetGoal)tempObj;
                       assetGoalComboBox.addItem(tempAssetGoal.getNameTextField());
```

```
                    }
                }
                //workspace
                tempObj = aScenario.scenarioManager.get(12);
                tempVec = (java.util.Vector)tempObj;
                posIndexFinalComboBox.removeAllItems();
//              for(int i = 0; i < tempVec.size(); i++)
//              {
                if(tempVec.size() > 0)
                {
                   tempObj = tempVec.get(0);
                   try
                   {
                      input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Workspace/"+tempObj.toString()));
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during input stream creation", "Exception",
                            JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      tempSet = (ScenarioElementSet)input.readObject();
                   }
                   catch(ClassNotFoundException classnotfoundException)
                   {
                      JOptionPane.showMessageDialog(this,
                            "Exception during file read - the object was not found",
                            "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file read",
                                  "Exception", JOptionPane.ERROR_MESSAGE);
                   }
                   try
                   {
                      input.close();
                   }
                   catch(IOException ioException)
                   {
                      JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                   }
//                 for(int j = 0; j < tempSet.elementContainer.size(); j++)
//                 {
                     Workspace tempWorkspace = new Workspace();
                     tempObj = tempSet.elementContainer.get(0);
                     tempWorkspace = (Workspace)tempObj;
                     DefaultListModel listModel = new DefaultListModel();
                     listModel = (DefaultListModel)tempWorkspace.getListModel();
                     for(int i = 0; i < listModel.size(); i++)
                     {
                        posIndexFinalComboBox.addItem(listModel.get(i).toString());
                     }
//                 }
//              }
                tempObj = tempVec = null;
                }
             }
```

```
public void Load()
{
}

public void Save()
{
}

private String dacGroupToString()
{
   String dacGroupString = new String();
   DefaultListModel listModel = new DefaultListModel();
   if(dac_added)
   {
      listModel = (DefaultListModel)dacList.getModel();
      dacGroupString +=
      "\t//List of DAC groups that the user belongs to\n"+
         "\tDACGroups:\n";
      for(int i = 0; i < listModel.getSize(); i++)
      {
         dacGroupString +=
         "\t\tPublic :end\n\t\t"+
         listModel.get(i).toString() +
         " :end\n";
      }
      dacGroupString += "\t:end  //DACGroups\n\n";
   }
   return dacGroupString;
}

//the following nametoString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller nametoString() methods are called from the primary
//asset toString() method.

//creates the asset goal list formatted output
private String assetGoalToString()
{
   Object tempObject = new Object();
   UserAssetGoal tempUAG = new UserAssetGoal();
   String uagString = new String();
   if(assetGoal_added)
   {
      uagString +=
         "\t//Asset goals specify what assets the user want s to access and what percentage\n"+
         "\t//of his time he wants to spend accessing them.\n\n"+
         "\t//Happiness is a scale factor determining how much of a hit to a user's happiness\n"+
         "\t//will occur if he can not fully accomplish this asset goal. \n\n"+
         "\t//Productivity is a scale factor determining how much of a hit to a user's \n"+
         "\t//productivity will occur if he can not fully accomplish this asset goal.\n\n";
      for(int i = 0; i < totalUserAssetGoalVec.size(); i++)
      {
         uagString +=
         "\tAssetGoal:\n";
         tempObject = totalUserAssetGoalVec.get(i);
         tempUAG = (UserAssetGoal)tempObject;
         uagString +=
         "\t\tAssetGoalName: "+tempUAG.getAssetGoal()+" :end\n"+
         "\t\tTargetUsage: "+tempUAG.getUsage()+" :end\n"+
         "\t\tHappiness: "+tempUAG.getHappiness()+" :end\n"+
         "\t\tProductivity: "+tempUAG.getProductivity()+" :end\n"+
```

```
            "\t:end  //of AssetGoal\n\n";
        }
    }
    return uagString;
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for an user descriptor
public String toString(String aText)
{
    String outputString = new String();
    outputString +=
    "User:"+
            "\t//User name string\n"+
            "\tName: "+nameTextField.getText()  +" :end\n\n"+
            "\t//A text string that specifies which department this user belongs to.\n"+
            "\tDept:  "+departmentFinalTextField.getText() +" :end\n\n"+
            "\tSupportSkills: \n"+
    "\t\tHardware: "+hardwareSkillsTextField.getText()+" :end\n"+
    "\t\tSoftware: "+softwareSkillsTextField.getText()+" :end\n"+
    "\t\tHumanInteraction: "+peopleSkillsTextField.getText()+" :end\n"+
            "\t:end //SupportSkills\n\n"+
            "\t//Number of days from beginning of scenario that user becomes available for hire\n"+
            "\t//Optional field. \n"+
            "\tAvailableDate: "+availableDateTextField.getText()+" :end\n\n"+
            "\t//The Secrecy clearance of this user.\n"+
            "\t//The user is authorized to read assets of this and lesser secrecy.\n"+
            "\tSecrecyClearance:  "+secrecyTextField.getText() +" :end\n\n"+
            "\t//The Integrity clearance of this user.\n"+
            "\t//The user is authorized to modify assets of this and lesser integrity.\n"+
            "\tIntegrityClearance:  "+integrityTextField.getText() +" :end\n\n"+
            dacGroupToString()+
            "\t//default to this DAC group\n"+
            "\tDefaultDAC: "+defaultDACTextField.getText()+" :end\n\n"+
            assetGoalToString()+
            "\t//A number used in the game to abstract how trustworthy the user is\n"+
            "\tTrustworthiness:  "+trustworthinessTextField.getText()+" :end\n\n"+
            "\t//Initial IT policy training level\n"+
            "\tInitialTraining: "+initialTrainingTextField.getText() +" :end\n\n"+
            "\t//Initial Happiness level of the user\n"+
            "\tHappiness: "+happinessTextField.getText()  +" :end\n\n"+
            "\t//initial productivity of the user\n"+
            "\tProductivity: "+productivityTextField.getText()  +" :end\n\n"+
            "\t//A number used by the game to abstract how skillful the user is.\n"+
            "\tSkill: "+skillTextField.getText() +" :end\n\n"+
            "\t//Which workspace is the user in\n"+
            "\tPosIndex: "+posIndexTextField.getText() +" :end\n\n"+
            "\t//how much does this user cost per month\n"+
            "\tCost: "+costTextField.getText() +" :end\n\n"+
            "\tGender: ";
    if(genderComboBox.getSelectedItem() != null)
    {
       outputString += genderComboBox.getSelectedItem().toString();
    }
    outputString += " :end\n\n"+
            "\t//A set of text string used in the game to describe this user to the player.\n"+
            "\tUserDescription: "+descriptionTextArea.getText()  +" :end\n\n"+
    ":end   //of User\n\n";
    return outputString;
}
```

```
public String getNameTextField()
{
   return nameTextField.getText();
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
   DefaultListModel listModel = new DefaultListModel();
   Object tempObject = new Object();
   UserAssetGoal tempUAG = new UserAssetGoal();
   for(int i = 0; i < totalUserAssetGoalVec.size(); i++)
   {
      tempObject = totalUserAssetGoalVec.get(i);
      tempUAG = (UserAssetGoal)temp Object;
      listModel.addElement(tempUAG.getUserAssetGoalString());
   }
   assetGoalList = null;
   assetGoalList = new JList(listModel);
   assetGoalList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   assetGoalScrollPane.setViewportView(assetGoalList);

   listModel = new DefaultListModel();
   for(int i = 0; i < mDACGroupVec.size(); i++)
   {
      listModel.addElement(mDACGroupVec.get(i));
   }
   dacList = null;
   dacList = new JList(listModel);
   dacList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   dacScrollPane.setViewportView(dacList);
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{

   dacAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         dacAddButtonActionPerformed(evt);
      }
   });

   dacRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         dacRemoveButtonActionPerformed(evt);
      }
   });

   addButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         addButtonActionPerformed(evt);
      }
   });
```

```
    removeButton.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            removeButtonActionPerformed(evt);
        }
    });
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario)
{
    populateDynamicSourceLists(aPath, aScenario);

    departmentComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
    departmentComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            departmentComboBoxActionPerformed(evt);
        }
    });


    secrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            secrecyComboBoxActionPerformed(evt);
        }
    });

    integrityComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            integrityComboBoxActionPerformed(evt);
        }
    });


    dacGroupComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            dacGroupComboBoxActionPerformed(evt);
        }
    });

    assetGoalComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            assetGoalComboBoxActionPerformed(evt);
        }
    });

    posIndexFinalComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            posIndexFinalComboBoxActionPerformed(evt);
        }
    });


    defaultDACComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
    defaultDACComboBox.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            defaultDACComboBoxActionPerformed(evt);
        }
    });
```

```
}

// Variables declaration - do not modify
private javax.swing.JButton addButton;
private javax.swing.JLabel agHappinessLabel;
private javax.swing.JTextField agHappinessTextField;
private javax.swing.JLabel agProductivityLabel;
private javax.swing.JTextField agProductivityTextField;
private javax.swing.JComboBox assetGoalComboBox;
private javax.swing.JLabel assetGoalLabel;
private javax.swing.JList assetGoalList;
private javax.swing.JPanel assetGoalPanel;
private javax.swing.JScrollPane assetGoalScrollPane;
private javax.swing.JTextField assetGroupFinalTextField;
private javax.swing.JLabel availableDateLabel;
private javax.swing.JTextField availableDateTextField;
private javax.swing.JLabel costLabel;
private javax.swing.JTextField costTextField;
private javax.swing.JButton dacAddButton;
private javax.swing.JComboBox dacGroupComboBox;
private javax.swing.JTextField dacGroupFinalTextField;
private javax.swing.JLabel dacGroupLabel;
private javax.swing.JPanel dacGroupPanel;
private javax.swing.JList dacList;
private javax.swing.JButton dacRemoveButton;
private javax.swing.JScrollPane dacScrollPane;
private javax.swing.JComboBox defaultDACComboBox;
private javax.swing.JLabel defaultDACLabel;
private javax.swing.JTextField defaultDACTextField;
private javax.swing.JComboBox departmentComboBox;
private javax.swing.JTextField departmentFinalTextField;
private javax.swing.JLabel departmentLabel;
private javax.swing.JLabel descriptionLabel;
private javax.swing.JScrollPane descriptionScrollPane;
private javax.swing.JTextArea descriptionTextArea;
private javax.swing.JComboBox genderComboBox;
private javax.swing.JLabel genderLabel;
private javax.swing.JLabel happinessLabel;
private javax.swing.JTextField happinessTextField;
private javax.swing.JLabel hardwareSkillsLabel;
private javax.swing.JTextField hardwareSkillsTextField;
private javax.swing.JLabel initialTrainingLabel;
private javax.swing.JTextField initialTrainingTextField;
private javax.swing.JComboBox integrityComboBox;
private javax.swing.JLabel integrityLabel;
private javax.swing.JTextField integrityTextField;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
private javax.swing.JLabel peopleSkillsLabel;
private javax.swing.JTextField peopleSkillsTextField;
private javax.swing.JComboBox posIndexFinalComboBox;
private javax.swing.JLabel posIndexLabel;
private javax.swing.JTextField posIndexTextField;
private javax.swing.JLabel productivityLabel;
private javax.swing.JTextField productivityTextField;
private javax.swing.JButton removeButton;
private javax.swing.JComboBox secrecyComboBox;
private javax.swing.JLabel secrecyLabel;
private javax.swing.JTextField secrecyTextField;
private javax.swing.JLabel skillLabel;
private javax.swing.JTextField skillTextField;
private javax.swing.JLabel softwareSkillsLabel;
```

```
    private javax.swing.JTextField softwareSkillsTextField;
    private javax.swing.JLabel targetUsageLabel;
    private javax.swing.JTextField targetUsageTextField;
    private javax.swing.JLabel trustworthinessLabel;
    private javax.swing.JTextField trustworthinessTextField;
    private javax.swing.JPanel userPanel;
    private javax.swing.JScrollPane userScrollPane;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private boolean dac_added;
    private java.util.Vector mDACGroupVec;
    private Scenario mScenario;
    private UserAssetGoal mUAG;
    private java.util.Vector totalUserAssetGoalVec;
    private boolean assetGoal_added;
}
```

AA. THE SOURCE OF: USER ASSET GOAL

```
/*
 * UserAssetGoal.java
 *
 * Created on February 15, 2004, 9:58 PM
 */

package CCSDT;

/**
 *
 * @author  KJohns
 *This class is used as an intermediate container to store user choices for
 *an asset goal in user.
 */
public class UserAssetGoal implements java.io.Serializable
{

    /** Creates a new instance of UserAssetGoal */
    public UserAssetGoal()
    {
        assetgoal = new String();
        usage = new String();
        happiness = new String();
        productivity = new String();
    }

    public void setAssetGoal(String aAssetGoal)
    {
        assetgoal = aAssetGoal;
    }
    public void setUsage(String aUsage)
    {
        usage = aUsage;
    }
    public void setHappiness(String aHappiness)
    {
        happiness = aHappiness;
    }
    public void setProductivity(String aProductivity)
    {
        productivity = aProductivity;
    }
    public String getAssetGoal()
```

```
    {
        return assetgoal;
    }
    public String getUsage()
    {
        return usage;
    }
    public String getHappiness()
    {
        return happiness;
    }
    public String getProductivity()
    {
        return productivity;
    }
    public String getUserAssetGoalString()
    {
        return "(ASSET GOAL)"+assetgoal + "(TARGET USAGE)" + usage + "(HAPPINESS)" + happiness +
"(PRODUCTIVITY)" + productivity;
    }

    private String assetgoal;
    private String usage;
    private String happiness;
    private String productivity;

}
```

BB. THE SOURCE OF: WORKSPACE

```
/*
 * Workspace.java
 *
 * Created on January 23, 2004, 10:10 AM
 */

package CCSDT;
import javax.swing.*;
/**
 *
 * @author  KJohns
 *
 *This class is the graphical representation of a Workspace descriptor.
 *This is where the Workspace form is managed and the Workspace
 *toString() resides.
 *
 */
public class Workspace extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Workspace */
    public Workspace()
    {
        initComponents();
        listVector = new java.util.Vector();
        nameTextField.setText(null);
    }


    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;
```

```
workspaceScrollPane = new javax.swing.JScrollPane();
workspacePanel = new javax.swing.JPanel();
nameLabel = new javax.swing.JLabel();
nameTextField = new javax.swing.JTextField();
listScrollPane = new javax.swing.JScrollPane();
listEntryList = new javax.swing.JList();
listEntryLabel = new javax.swing.JLabel();
listEntryTextField = new javax.swing.JTextField();
listEntryAddButton = new javax.swing.JButton();
listEntryRemoveButton = new javax.swing.JButton();

setLayout(new java.awt.GridLayout(1, 0));

workspacePanel.setLayout(new java.awt.GridBagLayout());

workspacePanel.setPreferredSize(new java.awt.Dimension(300, 420));
nameLabel.setText("Workspace List Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
workspacePanel.add(nameLabel, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
workspacePanel.add(nameTextField, gridBagConstraints);

listScrollPane.setPreferredSize(new java.awt.Dimension(250, 300));
listEntryList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
listScrollPane.setViewportView(listEntryList);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
workspacePanel.add(listScrollPane, gridBagConstraints);

listEntryLabel.setText("List Entry:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
workspacePanel.add(listEntryLabel, gridBagConstraints);

listEntryTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
workspacePanel.add(listEntryTextField, gridBagConstraints);

listEntryAddButton.setText("Add List Entry");
listEntryAddButton.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      listEntryAddButtonActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
workspacePanel.add(listEntryAddButton, gridBagConstraints);
```

```
        listEntryRemoveButton.setText("Remove List Entry");
        listEntryRemoveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                listEntryRemoveButtonActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        workspacePanel.add(listEntryRemoveButton, gridBagConstraints);

        workspaceScrollPane.setViewportView(workspacePanel);

        add(workspaceScrollPane);

    }
//the following listeners are for the buttons that either move or remove
    //entries from list boxes
    private void listEntryRemoveButtonActionPerformed(java.awt.event.ActionEvent evt) {
        if(listEntryList.getSelectedIndex() >= 0)
        {
            listVector.remove(listEntryList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)listEntryList.getModel();
            listModel.remove(listEntryList.getSelectedIndex());
            listEntryList = null;
            listEntryList = new JList(listModel);
            listEntryList.setSelectionMode(javax.swing.ListSelectionModel.
                                           SINGLE_SELECTION);
            listScrollPane.setViewportView(listEntryList);
        }
    }

    private void listEntryAddButtonActionPerformed(java.awt.event.ActionEvent evt) {
        listVector.addElement(listEntryTextField.getText());
        DefaultListModel listModel = new DefaultListModel();
        for(int i = 0; i < listVector.size(); i++)
        {
            listModel.addElement(listVector.get(i));
        }
        listEntryList = null;
        listEntryList = new JList(listModel);
        listEntryList.setSelectionMode(
                    javax.swing.ListSelectionModel.SINGLE_SELECTION);
        listScrollPane.setViewportView(listEntryList);
    }

    public void Load()
    {
    }

    public void Save()
    {
    }

    //used to get the list data
    public DefaultListModel getListModel()
    {
        return (DefaultListModel)listEntryList.getModel();
```

```
}

//Each descriptor form has a toString() method that is used by build() in
//the SDT to create the SDF.

//creates the formatted output for a workspace descriptor
//NOTE: workspace has no output in an SDF
public String toString(String aText)
{
   return null;
}

public String getNameTextField()
{
   return nameTextField.getText();
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
   DefaultListModel listModel = new DefaultListModel();
   for(int i = 0; i < listVector.size(); i++)
   {
      listModel.addElement(listVector.get(i));
   }
   listEntryList = null;
   listEntryList = new JList(listModel);
   listEntryList.setSelectionMode(
                  javax.swing.ListSelectionModel.SINGLE_SELECTION);
   listScrollPane.setViewportView(listEntryList);
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners()
{
   listEntryAddButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         listEntryAddButtonActionPerformed(evt);
      }
   });

   listEntryRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         listEntryRemoveButtonActionPerformed(evt);
      }
   });
}

// Variables declaration - do not modify
private javax.swing.JButton listEntryAddButton;
private javax.swing.JLabel listEntryLabel;
private javax.swing.JList listEntryList;
private javax.swing.JButton listEntryRemoveButton;
private javax.swing.JTextField listEntryTextField;
private javax.swing.JScrollPane listScrollPane;
private javax.swing.JLabel nameLabel;
private javax.swing.JTextField nameTextField;
```

```
    private javax.swing.JPanel workspacePanel;
    private javax.swing.JScrollPane workspaceScrollPane;
    // End of variables declaration
    private java.util.Vector listVector;
}
```

CC. THE SOURCE OF: ZONE

```
/*
 * Zone.java
 *
 * Created on January 7, 2004, 4:02 PM
 */

package CCSDT;
import javax.swing.*;
import java.io.*;
/**
 *
 * @author  kjohns
 *
 *This class is the graphical representation of a Zone descriptor.
 *This is where the Zone form is managed and the Zone
 *toString() resides.
 *
 */
public class Zone extends ScenarioElement implements java.io.Serializable
{

    /** Creates new form Zone */
    public Zone()
    {
        initComponents();
        procedure_added = false;
        permittedUsers_added = false;
        mProceduralSettingFileName = new String();
        mPermittedUsers_vector = new java.util.Vector();
        mNetworks_vector = new java.util.Vector();
    }

    private void initComponents() {
        java.awt.GridBagConstraints gridBagConstraints;

        alarmbuttonGroup = new javax.swing.ButtonGroup();
        locksbuttonGroup = new javax.swing.ButtonGroup();
        zoneScrollPane = new javax.swing.JScrollPane();
        zonePanel = new javax.swing.JPanel();
        textAndComboPanel = new javax.swing.JPanel();
        nameLabel = new javax.swing.JLabel();
        siteNameLabel = new javax.swing.JLabel();
        ulcXLabel = new javax.swing.JLabel();
        lrcXLabel = new javax.swing.JLabel();
        ulcYTextField = new javax.swing.JTextField();
        ulcXTextField = new javax.swing.JTextField();
        siteNameTextField = new javax.swing.JTextField();
        nameTextField = new javax.swing.JTextField();
        proceduralSettingsLabel = new javax.swing.JLabel();
        proceduralSettingsComboBox = new javax.swing.JComboBox();
        proceduralSettingsFinalTextField = new javax.swing.JTextField();
        lrcXTextField = new javax.swing.JTextField();
        lrcYTextField = new javax.swing.JTextField();
        ulcYLabel = new javax.swing.JLabel();
```

```
lrcYLabel = new javax.swing.JLabel();
constraintsPanel = new javax.swing.JPanel();
clerancePanel = new javax.swing.JPanel();
zoneSecrecyLabel = new javax.swing.JLabel();
zoneIntegrityLabel = new javax.swing.JLabel();
zoneIntegrityComboBox = new javax.swing.JComboBox();
zoneSecrecyComboBox = new javax.swing.JComboBox();
zoneSecrecyTextField = new javax.swing.JTextField();
zoneIntegrityFinalTextField = new javax.swing.JTextField();
boolPanel = new javax.swing.JPanel();
perimeterAlarmPanel = new javax.swing.JPanel();
expensiveAlarmRadioButton = new javax.swing.JRadioButton();
moderateAlarmRadioButton = new javax.swing.JRadioButton();
noAlarmRadioButton = new javax.swing.JRadioButton();
doorLockPanel = new javax.swing.JPanel();
cipherLockRadioButton = new javax.swingJRadioButton();
expensiveRetinalRadioButton = new javax.swing.JRadioButton();
moderateRetinalRadioButton = new javax.swing.JRadioButton();
keyLockRadioButton = new javax.swing.JRadioButton();
noLockRadioButton = new javax.swing.JRadioButton();
checkboxPanel = new javax.swing.JPanel();
receptionishCheckBox = new javax.swing.JCheckBox();
doorGuardCheckBox = new javax.swing.JCheckBox();
patrolGuardCheckBox = new javax.swing.JCheckBox();
noMediaCheckBox = new javax.swing.JCheckBox();
noPhoneCheckBox = new javax.swing.JCheckBox();
reinforcedWallCheckBox = new javax.swing.JCheckBox();
surveillanceCameasCheckBox = new javax.swing.JCheckBox();
escortedVisitorsCheckBox = new javax.swing.JCheckBox();
visualInspectionCheckBox = new javax.swing.JCheckBox();
xRayCheckBox = new javax.swing.JCheckBox();
idBadgesAlwaysCheckBox = new javax.swing.JCheckBox();
listsPanel = new javax.swing.JPanel();
permittedUsersMoveButton = new javax.swing.JButton();
permittedUsersRemoveButton = new javax.swing.JButton();
permittedUsersFinalScrollPane = new javax.swing.JScrollPane();
permittedUsersFinalList = new javax.swing.JList();
permittedUsersLabel = new javax.swing.JLabel();
permittedUsersSourceScrollPane = new javax.swing.JScrollPane();
permittedUsersSourceList = new javax.swing.JList();

setLayout(new java.awt.GridLayout(1, 0));

zonePanel.setLayout(new java.awt.GridBagLayout());

zonePanel.setPreferredSize(new java.awt.Dimension(765, 791));
textAndComboPanel.setLayout(new java.awt.GridBagLayout());

nameLabel.setLabelFor(nameTextField);
nameLabel.setText("Zone Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(nameLabel, gridBagConstraints);

siteNameLabel.setText("Site Name:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(siteNameLabel, gridBagConstraints);

ulcXLabel.setText("Zone Upper Left Corner X value:");
```

```
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(ulcXLabel, gridBagConstraints);

lrcXLabel.setText("Zone Lower Right Corner X value:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(lrcXLabel, gridBagConstraints);

ulcYTextField.setPreferredSize(new java.awt.Dimension(65, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(ulcYTextField, gridBagConstraints);

ulcXTextField.setPreferredSize(new java.awt.Dimension(65, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(ulcXTextField, gridBagConstraints);

siteNameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(siteNameTextField, gridBagConstraints);

nameTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
text AndComboPanel.add(nameTextField, gridBagConstraints);

proceduralSettingsLabel.setText("Procedural Settings:");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
textAndComboPanel.add(proceduralSettingsLabel, gridBagConstraints);

proceduralSettingsComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
proceduralSettingsComboBox.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      proceduralSettingsComboBoxActionPerformed(evt);
   }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
textAndComboPanel.add(proceduralSettingsComboBox, gridBagConstraints);

proceduralSettingsFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
```

```
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 2;
                gridBagConstraints.gridy = 3;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                textAndComboPanel.add(proceduralSettingsFinalTextField, gridBagConstraints);

                lrcXTextField.setPreferredSize(new java.awt.Dimension(65, 20));
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 3;
                gridBagConstraints.gridy = 1;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                textAndComboPanel.add(lrcXTextField, gridBagConstraints);

                lrcYText Field.setPreferredSize(new java.awt.Dimension(65, 20));
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 3;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
                textAndComboPanel.add(lrcYTextField, gridBagConstraints);

                ulcYLabel.setText("Zone Upper Left Corner Y Value:");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 0;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
                textAndComboPanel.add(ulcYLabel, gridBagConstraints);

                lrcYLabel.setText("Zone Lower Right Corner Y Value:");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 2;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
                textAndComboPanel.add(lrcYLabel, gridBagConstraints);

                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
                zonePanel.add(textAndComboPanel, gridBagConstraints);

                constraintsPanel.setLayout(new java.awt.GridBagLayout());

                constraintsPanel.setBorder(new javax.swing.border.TitledBorder("Constraints on User Access to
Zone"));
                clerancePanel.setLayout(new java.awt.GridBagLayout());

                clerancePanel.setBorder(new javax.swing.border.TitledBorder("Clearance Based Constraints"));
                zoneSecrecyLabel.setText("Zone Secrecy Level:");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 2;
                gridBagConstraints.gridy = 1;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
                clerancePanel.add(zoneSecrecyLabel, gridBagConstraints);

                zoneIntegrityLabel.setText("Zone Integrity Level:");
                gridBagConstraints = new java.awt.GridBagConstraints();
                gridBagConstraints.gridx = 2;
                gridBagConstraints.gridy = 2;
                gridBagConstraints.anchor = java.awt.GridBagConstraints.EAST;
                clerancePanel.add(zoneIntegrityLabel, gridBagConstraints);

                zoneIntegrityComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
                zoneIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
                    public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        zoneIntegrityComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 2;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
clerancePanel.add(zoneIntegrityComboBox, gridBagConstraints);

zoneSecrecyComboBox.setPreferredSize(new java.awt.Dimension(150, 20));
zoneSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        zoneSecrecyComboBoxActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
clerancePanel.add(zoneSecrecyComboBox, gridBagConstraints);

zoneSecrecyTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 1;
clerancePanel.add(zoneSecrecyTextField, gridBagConstraints);

zoneIntegrityFinalTextField.setPreferredSize(new java.awt.Dimension(150, 20));
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 4;
gridBagConstraints.gridy = 2;
clerancePanel.add(zoneIntegrityFinalTextField, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.insets = new java.awt.Insets(20, 0, 0, 0);
constraintsPanel.add(clerancePanel, gridBagConstraints);

boolPanel.setLayout(new java.awt.GridBagLayout());

boolPanel.setBorder(new javax.swing.border.TitledBorder("Physical Security Settings"));
boolPanel.setPreferredSize(new java.awt.Dimension(450, 290));
perimeterAlarmPanel.setLayout(new java.awt.GridBagLayout());

perimeterAlarmPanel.setBorder(new javax.swing.border.TitledBorder("Perimeter Alarm Settings"));
perimeterAlarmPanel.setPreferredSize(new java.awt.Dimension(225, 97));
expensiveAlarmRadioButton.setSelected(true);
expensiveAlarmRadioButton.setText("Expensive");
alarmbuttonGroup.add(expensiveAlarmRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 9;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
perimeterAlarmPanel.add(expensiveAlarmRadioButton, gridBagConstraints);

moderateAlarmRadioButton.setText("Moderate");
alarmbuttonGroup.add(moderateAlarmRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
```

```
gridBagConstraints.gridy = 10;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
perimeterAlarmPanel.add(moderateAlarmRadioButton, gridBagConstraints);

noAlarmRadioButton.setText("No alarm");
alarmbuttonGroup.add(noAlarmRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 11;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
perimeterAlarmPanel.add(noAlarmRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 0;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel.add(perimeterAlarmPanel, gridBagConstraints);

doorLockPanel.setLayout(new java.awt.GridBagLayout());

doorLockPanel.setBorder(new javax.swing.border.TitledBorder("Door Lock Settings"));
doorLockPanel.setPreferredSize(new java.awt.Dimension(225, 145));
cipherLockRadioButton.setSelected(true);
cipherLockRadioButton.setText("Cipher locks");
locksbuttonGroup.add(cipherLockRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 13;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
doorLockPanel.add(cipherLockRadioButton, gridBagConstraints);

expensiveRetinalRadioButton.setText("Expensive retinal scanner");
locksbuttonGroup.add(expensiveRetinalRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 14;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
doorLockPanel.add(expensiveRetinalRadioButton, gridBagConstraints);

moderateRetinalRadioButton.setText("Moderate retinal scanner");
locksbuttonGroup.add(moderateRetinalRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 15;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
doorLockPanel.add(moderateRetinalRadioButton, gridBagConstraints);

keyLockRadioButton.setText("Key locks");
locksbuttonGroup.add(keyLockRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 16;
```

```
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
doorLockPanel.add(keyLockRadioButton, gridBagConstraints);

noLockRadioButton.setText("No locks");
locksbuttonGroup.add(noLockRadioButton);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 17;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(0, 25, 0, 0);
doorLockPanel.add(noLockRadioButton, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx= 1;
gridBagConstraints.gridy = 1;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
boolPanel.add(doorLockPanel, gridBagConstraints);

checkboxPanel.setLayout(new java.awt.GridBagLayout());

receptionishCheckBox.setText("Receptionist");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 7;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(receptionishCheckBox, gridBagConstraints);

doorGuardCheckBox.setText("Guard at door");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 8;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(doorGuardCheckBox, gridBagConstraints);

patrolGuardCheckBox.setText("Patrolling guard");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 9;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(patrolGuardCheckBox, gridBagConstraints);

noMediaCheckBox.setText("Prohibit media");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 10;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(noMediaCheckBox, gridBagConstraints);

noPhoneCheckBox.setText("Prohibit phone devices");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 11;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(noPhoneCheckBox, gridBagConstraints);

reinforcedWallCheckBox.setText("Reinforced walls");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 12;
```

```
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(reinforcedWallCheckBox, gridBagConstraints);

surveillanceCameasCheckBox.setText("Surveillance cameras");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 13;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(surveillanceCameasCheckBox, gridBagConstraints);

escortedVisitorsCheckBox.setText("Permit escorted visitors");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 14;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(escortedVisitorsCheckBox, gridBagConstraints);

visualInspectionCheckBox.setText("Visual inspection of all personnel");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 15;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(visualInspectionCheckBox, gridBagConstraints);

xRayCheckBox.setText("X-Ray package");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 16;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(xRayCheckBox, gridBagConstraints);

idBadgesAlwaysCheckBox.setText("ID badges worn at all times");
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 17;
gridBagConstraints.anchor = java.awt.GridBagConstraints.WEST;
checkboxPanel.add(idBadgesAlwaysCheckBox, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 0;
gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
boolPanel.add(checkboxPanel, gridBagConstraints);

gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 2;
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.insets = new java.awt.Insets(20, 0, 0, 0);
constraintsPanel.add(boolPanel, gridBagConstraints);

listsPanel.setLayout(new java.awt.GridBagLayout());

listsPanel.setBorder(new javax.swing.border.TitledBorder("User Based Constraints"));
permittedUsersMoveButton.setText(">>>>>");
permittedUsersMoveButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        permittedUsersMoveButtonActionPerformed(evt);
    }
});

gridBagConstraints = new java.awt.GridBagConstraints();
```

```
        gridBagConstraints.gridx = 2;
        gridBagConstraints.gridy = 4;
        listsPanel.add(permittedUsersMoveButton, gridBagConstraints);

        permittedUsersRemoveButton.setText("Remove");
        permittedUsersRemoveButton.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                permittedUsersRemoveButtonActionPerformed(evt);
            }
        });

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 5;
        listsPanel.add(permittedUsersRemoveButton, gridBagConstraints);

        permittedUsersFinalScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
        permittedUsersFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        permittedUsersFinalScrollPane.setViewportView(permittedUsersFinalList);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 4;
        listsPanel.add(permittedUsersFinalScrollPane, gridBagConstraints);

        permittedUsersLabel.setText("Permitted Users:");
        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = java.awt.GridBagConstraints.NORTHEAST;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        listsPanel.add(permittedUsersLabel, gridBagConstraints);

        permittedUsersSourceScrollPane.setPreferredSize(new java.awt.Dimension(150, 150));
        permittedUsersSourceList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
        permittedUsersSourceScrollPane.setViewportView(permittedUsersSourceList);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        listsPanel.add(permittedUsersSourceScrollPane, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
        gridBagConstraints.insets = new java.awt.Insets(20, 0, 0, 0);
        constraintsPanel.add(listsPanel, gridBagConstraints);

        gridBagConstraints = new java.awt.GridBagConstraints();
        gridBagConstraints.insets = new java.awt.Insets(10, 0, 0, 0);
        zonePanel.add(constraintsPanel, gridBagConstraints);

        zoneScrollPane.setViewportView(zonePanel);

        add(zoneScrollPane);

    }
//the following listeners are for the buttons that either move or remove
    //entries from list boxes
    private void permittedUsersRemoveButtonActionPerformed(java.awt.event.ActionEvent evt)
```

```
    {
        if(permittedUsersFinalList.getSelectedIndex() >= 0)
        {
            mPermittedUsers_vector.remove(permittedUsersFinalList.getSelectedIndex());
            DefaultListModel listModel = new DefaultListModel();
            listModel = (DefaultListModel)permittedUsersFinalList.getModel();
            listModel.remove(permittedUsersFinalList.getSelectedIndex());
            permittedUsersFinalList = new JList(listModel);
            permittedUsersFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            permittedUsersFinalScrollPane.setViewportView(permittedUsersFinalList);
        }
    }

    private void permittedUsersMoveButtonActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(permittedUsersSourceList.getSelectedValue() != null)
        {
            Object tempObject = new Object();
            mPermittedUsers_vector.addElement(permittedUsersSourceList.getSelectedValue().toString());
            DefaultListModel listModel = new DefaultListModel();
            for(int i = 0; i < mPermittedUsers_vector.size(); i++)
            {
                tempObject = mPermittedUsers_vector.get(i);
                listModel.addElement(tempObject);
            }
            permittedUsersFinalList = null;
            permittedUsersFinalList = new JList(listModel);
            permittedUsersFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
            permittedUsersFinalScrollPane.setViewportView(permittedUsersFinalList);
            permittedUsers_added = true;
        }
    }
//the following listeners set a text field based on a combo box selection
    private void zoneIntegrityComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(zoneIntegrityComboBox.getSelectedItem() != null)
        {
            zoneIntegrityFinalTextField.setText(
                    zoneIntegrityComboBox.getSelectedItem().toString());
        }
    }

    private void zoneSecrecyComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(zoneSecrecyComboBox.getSelectedItem() != null)
        {
            zoneSecrecyTextField.setText(
                        zoneSecrecyComboBox.getSelectedItem().toString());
        }
    }

    private void proceduralSettingsComboBoxActionPerformed(java.awt.event.ActionEvent evt)
    {
        if(proceduralSettingsComboBox.getSelectedIndex() >= 0)
        {
            proceduralSettingsFinalTextField.setText(
                proceduralSettingsElementListVec.get(
                        proceduralSettingsComboBox.getSelectedIndex()).
                            toString());
            mProceduralSettingFileName = proceduralSettingsFileListVec.get(
                  proceduralSettingsComboBox.getSelectedIndex()).toString();
            procedure_added = true;
```

```
    }
  }

  //Interface combo box and list box content comes from one of two sources
  //the content is either from ini files or from reusable sets that have
  //already been added to the scenario in development.
  //If the content come from ini files it is designated static.  This is
  //because the content of ini files does not change.
  //If the content comes from reusable sets that have been added to the
  //scenario in development it is designated dynamic.  This is because
  //the content may be empty, of any lenght and change frequently

  //populateDynamicSourceLists() takes datapath and startupScenario as
  //parameters in ScenarioDefinitionTool.java and is used to add dynamic
  //content to combo boxes and list boxes.
  //First the scenairo is used to get the added reusable sets
  //Second any combo boxes to be used are cleared and reinitialized
  //Third for each reusable set added a file input object is created
  //Fourth for each member of the set the element name is added to the
  //combo box or list.
  public void populateDynamicSourceLists(String aPath, Scenario aScenario)
  {
    //uses the Scenario passed to it to set the dynamic values of
    //dependent sets
    //ProceduralSettings
    //Secrecy
    //Integrity
    //User
    //Network
    mScenario = aScenario;
    Object tempObj = new Object();
    java.util.Vector tempVec = new java.util.Vector();
    ScenarioElementSet tempSet = new ScenarioElementSet();
    java.util.Vector networksListVec = new java.util.Vector();
    proceduralSettingsFileListVec = new java.util.Vector();
    proceduralSettingsElementListVec = new java.util.Vector();
    userListVec = new java.util.Vector();

    //ProceduralSettings
    tempObj = aScenario.scenarioManager.get(14);
    tempVec = (java.util.Vector)tempObj;
    tempObj = tempVec.get(1);//get the proc settings
    tempVec = (java.util.Vector)tempObj;
    proceduralSettingsComboBox.removeAllItems();
    for(int i = 0; i < tempVec.size(); i++)
    {
      tempObj = tempVec.get(i);
      try
      {
        input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural
Settings/"+tempObj.toString()));
      }
      catch(IOException ioException)
      {
        JOptionPane.showMessageDialog(this,
            "Exception during input stream creation", "Exception",
            JOptionPane.ERROR_MESSAGE);
      }
      try
      {
        tempSet = (ScenarioElementSet)input.readObject();
```

```
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                       "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       //save the set filename for later
                       proceduralSettingsFileListVec.addElement(tempSet.getelementSetName());
                       tempObj = tempSet.elementContainer.get(j);
                       ProceduralSettings tempProcSettings = new ProceduralSettings();
                       tempProcSettings = (ProceduralSettings)tempObj;
                       //save the element name for later
                       proceduralSettingsElementListVec.addElement(tempProcSettings.getNameTextField());
                       proceduralSettingsComboBox.addItem(tempProcSettings.getNameTextField());
                    }
                 }
                 //Secrecy
                 tempObj = aScenario.scenarioManager.get(9);
                 tempVec = (java.util.Vector)tempObj;
                 zoneSecrecyComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Secrecy/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during input stream creation", "Exception",
                                JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
```

```
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                       "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       Secrecy tempSecrecy = new Secrecy();
                       tempObj = tempSet.elementContainer.get(j);
                       tempSecrecy = (Secrecy)tempObj;
                       zoneSecrecyComboBox.addItem(tempSecrecy.getNameTextField());
                    }
                 }
                 //Integrity
                 tempObj = aScenario.scenarioManager.get(6);
                 tempVec = (java.util.Vector)tempObj;
                 zoneIntegrityComboBox.removeAllItems();
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/Integrity/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                     "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                       "Exception", JOptionPane.ERROR_MESSAGE);
```

```
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       Integrity tempIntegrity = new Integrity();
                       tempObj = tempSet.elementContainer.get(j);
                       tempIntegrity = (Integrity)tempObj;
                       zoneIntegrityComboBox.addItem(tempIntegrity.getNameTextField());
                    }
                 }
                 //User
                 tempObj = aScenario.scenarioManager.get(11);
                 tempVec = (java.util.Vector)tempObj;
                 for(int i = 0; i < tempVec.size(); i++)
                 {
                    tempObj = tempVec.get(i);
                    try
                    {
                       input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aPath+"CyberCIEGE/SDT/Reusable Sets Library/User/"+tempObj.toString()));
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during input stream creation", "Exception",
                             JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                             "Exception during file read - the object was not found",
                             "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read",
                                   "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                      "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       User tempUser = new User();
                       tempObj = tempSet.elementContainer.get(j);
                       tempUser = (User)tempObj;
                       userListVec.add(tempUser.getNameTextField());
                    }
                 }
                 permittedUsersSourceList.setListData(userListVec);
                 tempObj = tempVec = null;
              }
```

```
public void Load()
{
}

public void Save()
{
}

//the following name_toString() methods below are used to create the
//nested lists that appear in the Asset descriptor of an SDF.
//These smaller name_toString() methods are called from the primary
//asset toString() method.

//creates the permitted users list formatted output
private String permittedUsersToString()
{
   String permittedUsersString = new String();
   DefaultListModel listModel = new DefaultListModel();
   if(permittedUsers_added)
   {
      listModel = (DefaultListModel)permittedUsersFinalList.getModel();
      permittedUsersString +=
      "\n\t//The tagname of a physical permitted users definiton\n";
      for(int i = 0; i < listModel.getSize(); i++)
      {
         permittedUsersString +=
         "\tPermittedUsers: " +
         listModel.get(i).toString() +
         " :end\n";
      }
   }
   return permittedUsersString;
}

public String toString(String aText)
{
   String outputString = new String();
   outputString +=
   "Zone:\n"+
   "\tName: "+nameTextField.getText()+" :end\n\n"+
         "\tSite: "+siteNameTextField.getText()+" :end\n\n"+
   "\t//The following procedural security component  defaults act as initial defaults for\n"+
   "\t//components created in this zone. They are initially associated with, but not \n"+
   "\t//enforced  by components.   They reflect procedural based user actions, and are\n"+
   "\t//therefore very dependent on the level of user training.\n\n";
   if(procedure_added)
   {
      Object tempObj = new Object();
      ScenarioElementSet tempSet = new ScenarioElementSet();
      try
      {
         input = new java.io.ObjectInputStream(new
java.io.FileInputStream(aText+"CyberCIEGE/SDT/Reusable Sets Library/Other/Procedural
Settings/"+mProceduralSettingFileName));
      }
      catch(IOException ioException)
      {
         JOptionPane.showMessageDialog(this,
                  "Exception during input stream creation", "Exception",
                  JOptionPane.ERROR_MESSAGE);
      }
      try
```

```
                    {
                       tempSet = (ScenarioElementSet)input.readObject();
                    }
                    catch(ClassNotFoundException classnotfoundException)
                    {
                       JOptionPane.showMessageDialog(this,
                                "Exception during file read - the object was not found",
                                "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file read-zone tostring",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    try
                    {
                       input.close();
                    }
                    catch(IOException ioException)
                    {
                       JOptionPane.showMessageDialog(this, "Exception during file close",
                                    "Exception", JOptionPane.ERROR_MESSAGE);
                    }
                    for(int j = 0; j < tempSet.elementContainer.size(); j++)
                    {
                       tempObj = tempSet.elementContainer.get(j);
                       ProceduralSettings tempProcSettings = new ProceduralSettings();
                       tempProcSettings = (ProceduralSettings)tempObj;

if(tempProcSettings.getNameTextField().toString().equalsIgnoreCase(proceduralSettingsFinalTextField.getText()))
                       {
                          outputString += tempProcSettings.toString(null);
                       }
                    }
                 }
                 outputString +=
                 "\n\n\t//END of procedural security component defaults\n\n"+
                 "\t//All doors in a zone are assigned the same physical security properties at\n"+
                 "\t//startup as all other doors in that zone. However, there could also be open access\n"+
                 "\t//to the zone (i.e. a large open space connecting it to another zone) in addition to\n"+
                 "\t//the doors, and these  open area connections will of course not have the same\n"+
                 "\t//physical security properties as the doors.\n\n"+
                 "\t//Receptionist must be on staff.\n"+
                 "\t//Receptionist efficiency subject to bribery, social engineering, and narcolepsy.\n"+
                 "\tReceptionist: "+String.valueOf(receptionishCheckBox.isSelected())+" :end\n\n"+
                 "\t//Guard at doors must be on staff.\n"+
                 "\t//Subject to bribery, social engineering, and sometimes narcolepsy.\n"+
                 "\tGuardAtDoor: "+String.valueOf(doorGuardCheckBox.isSelected())+" :end\n\n"+
                 "\t//Patrolling guards must be on staff.\n"+
                 "\t//Subject to bribery, social engineering\n"+
                 "\tPatrollingGuard: "+String.valueOf(patrolGuardCheckBox.isSelected())+" :end\n\n"+
                 "\t//Prohibit media from leaving.  Effectiveness depends on receptionist or\n"+
                 "\t//guard, VisualPeopleInspection and XrayPackages.\n"+
                 "\tProhibitMedia: "+String.valueOf(noMediaCheckBox.isSelected())+" :end\n\n"+
                 "\t//Prohibit communication devices from entering.  Effectiveness depends on\n"+
                 "\t//receptionist or guard, VisualPeopleInspection and XrayPackages.\n"+
                 "\tProhibitPhoneDevices: "+String.valueOf(noPhoneCheckBox.isSelected())+" :end\n\n"+
                 "\t//Expensive Electronic Alarms around perimeter.\n"+
                 "\tExpensivePerimeterAlarms: "+String.valueOf(expensiveAlarmRadioButton.isSelected())+"
:end\n\n"+
                 "\t//Moderate priced electronic alarms around perimeter\n"+
                 "\tModeratePerimeterAlarms: "+String.valueOf(moderateAlarmRadioButton.isSelected())+" :end\n\n"+
```

```
   "\t//Re-enforced walls\n"+
   "\tRe-enforcedWalls: "+String.valueOf(reinforcedWallCheckBox.isSelected())+" :end\n\n"+
   "\t//Surveillance cameras\n"+
   "\tSurveillanceCameras: "+String.valueOf(surveillanceCameasCheckBox.isSelected())+" :end\n\n"+
   "\t//Permit authorized users to escort unauthorized users.  Lack of this creates\n"+
   "\t//inefficiency and user unhappiness.\n"+
   "\tPermitEscortedVisitors: "+String.valueOf(escortedVisitorsCheckBox.isSelected())+" :end\n\n"+
   "\t//Visual and hand-bag search packages and people entering and leaving\n"+
   "\t//Affects effectiveness of ProhibitMedia and ProhibitPhoneDevices.  Needs door\n"+
   "\t//guard or receptionist to implement.\n"+
   "\tVisualPeopleInspection: "+String.valueOf(visualInspectionCheckBox.isSelected())+" :end\n\n"+
   "\t//Xray packages\n"+
   "\t//Affects effectiveness of ProhibitMedia and ProhibitPhoneDevices.  Needs door\n"+
   "\t//guard or receptionist to implement.\n"+
   "\tXrayPackages: "+String.valueOf(xRayCheckBox.isSelected())+" :end\n\n"+
   "\t//Key lock on door.  Users lose keys.\n"+
   "\t//Vandals plug keyholes.  Attackers can duplicate keys.\n"+
   "\tKeyLockOnDoor: "+String.valueOf(keyLockRadioButton.isSelected())+" :end\n\n"+
   "\t//Cipher lock on door.  Users can forget codes.  Attackers can employ shortcuts\n"+
   "\t//and tricks like colored chalk.\n"+
   "\tCipherLockOnDoor: "+String.valueOf(cipherLockRadioButton.isSelected())+" :end\n\n"+
   "\t//Expensive iris scanner.   Few false negatives.\n"+
   "\tExpensiveIrisScanner: "+String.valueOf(expensiveRetinalRadioButton.isSelected())+" :end\n\n"+
   "\t//Moderate priced iris scanner.  Bloodshot eyes create false negatives.\n"+
   "\tModerateIrisScanner: "+String.valueOf(moderateRetinalRadioButton.isSelected())+" :end\n\n"+
   "\t//Badges required for everyone in the zone.   Some users forget badges\n"+
   "\t//resulting in inefficiency. Some users just unhappy about having to wear badges.\n"+
   "\tBadges: "+String.valueOf(idBadgesAlwaysCheckBox.isSelected())+" :end\n\n"+
   "\t//This is the intended Secrecy Level for people entering this zone.\n"+
   "\tSecrecy: "+zoneSecrecyTextField.getText()+" :end\n\n"+
   "\t//This is the intended Integrity Level for people entering this zone.\n"+
   "\tIntegrity: "+zoneIntegrityFinalTextField.getText()+" :end\n\n"+
   permittedUsersToString()+
   "\t//These are the upper left corner and lower right corner coordinates\n"+
   "\tULC:  "+ulcXTextField.getText()+" "+ulcYTextField.getText()+" :end // Integer values\n"+
   "\tLRC:  "+lrcXTextField.getText()+" "+lrcYTextField.getText()+" :end\n\n"+
   ":end    //of Zone\n\n";
   return outputString;
}

public String getNameTextField()
{
   return nameTextField.getText();
}

//After save operations cause the list boxes in forms to be cleared
//this method repopulates those list boxes and reinitilizes them.
//The code in this method is literally taken line for line from the
//button listeners above.
public void reInitLists()
{
   Object tempObject = new Object();
   DefaultListModel listModel = new DefaultListModel();
   for(int i = 0; i < mPermittedUsers_vector.size(); i++)
   {
      tempObject = mPermittedUsers_vector.get(i);
      listModel.addElement(tempObject);
   }
   permittedUsersFinalList = null;
   permittedUsersFinalList = new JList(listModel);
   permittedUsersFinalList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
   permittedUsersFinalScrollPane.setViewportView(permittedUsersFinalList);
```

```
}

//This method provides a way to reinitialize the button listeners when
//they die after IO operations.  The code is taken line for line from the
//initComponents() method.
public void reInitButtons()
{
   permittedUsersMoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         permittedUsersMoveButtonActionPerformed(evt);
      }
   });

   permittedUsersRemoveButton.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         permittedUsersRemoveButtonActionPerformed(evt);
      }
   });
}

//This method provides a way to reinitialize any miscellaneous
//(non-list, non-button) listeners when they die after IO operations.
//The code is taken line for line from  the initComponents() method.
public void reInitListeners(String aPath, Scenario aScenario)
{
   populateDynamicSourceLists(aPath, aScenario);

   zoneIntegrityComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         zoneIntegrityComboBoxActionPerformed(evt);
      }
   });

   zoneSecrecyComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         zoneSecrecyComboBoxActionPerformed(evt);
      }
   });

   proceduralSettingsComboBox.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
         proceduralSettingsComboBoxActionPerformed(evt);
      }
   });
}

// Variables declaration - do not modify
private javax.swing.ButtonGroup alarmbuttonGroup;
private javax.swing.JPanel boolPanel;
private javax.swing.JPanel checkboxPanel;
private javax.swing.JRadioButton cipherLockRadioButton;
private javax.swing.JPanel clerancePanel;
private javax.swing.JPanel constraintsPanel;
private javax.swing.JCheckBox doorGuardCheckBox;
private javax.swing.JPanel doorLockPanel;
private javax.swing.JCheckBox escortedVisitorsCheckBox;
private javax.swing.JRadioButton expensiveAlarmRadioButton;
private javax.swing.JRadioButton expensiveRetinalRadioButton;
private javax.swing.JCheckBox idBadgesAlwaysCheckBox;
private javax.swing.JRadioButton keyLockRadioButton;
```

```
    private javax.swing.JPanel listsPanel;
    private javax.swing.ButtonGroup locksbuttonGroup;
    private javax.swing.JLabel lrcXLabel;
    private javax.swing.JTextField lrcXTextField;
    private javax.swing.JLabel lrcYLabel;
    private javax.swing.JTextField lrcYTextField;
    private javax.swing.JRadioButton moderateAlarmRadioButton;
    private javax.swing.JRadioButton moderateRetinalRadioButton;
    private javax.swing.JLabel nameLabel;
    private javax.swing.JTextField nameTextField;
    private javax.swing.JRadioButton noAlarmRadioButton;
    private javax.swing.JRadioButton noLockRadioButton;
    private javax.swing.JCheckBox noMediaCheckBox;
    private javax.swing.JCheckBox noPhoneCheckBox;
    private javax.swing.JCheckBox patrolGuardCheckBox;
    private javax.swing.JPanel perimeterAlarmPanel;
    private javax.swing.JList permittedUsersFinalList;
    private javax.swing.JScrollPane permittedUsersFinalScrollPane;
    private javax.swing.JLabel permittedUsersLabel;
    private javax.swing.JButton permittedUsersMoveButton;
    private javax.swing.JButton permittedUsersRemoveButton;
    private javax.swing.JList permittedUsersSourceList;
    private javax.swing.JScrollPane permittedUsersSourceScrollPane;
    private javax.swing.JComboBox proceduralSettingsComboBox;
    private javax.swing.JTextField proceduralSettingsFinalTextField;
    private javax.swing.JLabel proceduralSettingsLabel;
    private javax.swing.JCheckBox receptionishCheckBox;
    private javax.swing.JCheckBox reinforcedWallCheckBox;
    private javax.swing.JLabel siteNameLabel;
    private javax.swing.JTextField siteNameTextField;
    private javax.swing.JCheckBox surveillanceCameasCheckBox;
    private javax.swing.JPanel textAndComboPanel;
    private javax.swing.JLabel ulcXLabel;
    private javax.swing.JTextField ulcXText Field;
    private javax.swing.JLabel ulcYLabel;
    private javax.swing.JTextField ulcYTextField;
    private javax.swing.JCheckBox visualInspectionCheckBox;
    private javax.swing.JCheckBox xRayCheckBox;
    private javax.swing.JComboBox zoneIntegrityComboBox;
    private javax.swing.JTextField zoneIntegrityFinalTextField;
    private javax.swing.JLabel zoneIntegrityLabel;
    private javax.swing.JPanel zonePanel;
    private javax.swing.JScrollPane zoneScrollPane;
    private javax.swing.JComboBox zoneS ecrecyComboBox;
    private javax.swing.JLabel zoneSecrecyLabel;
    private javax.swing.JTextField zoneSecrecyTextField;
    // End of variables declaration
    transient private java.io.ObjectInputStream input;
    private Scenario mScenario;
    private boolean procedure_added;
    private boolean permittedUsers_added;
    private String mProceduralSettingFileName;
    private java.util.Vector mPermittedUsers_vector;
    private java.util.Vector mNetworks_vector;
    private java.util.Vector proceduralSettingsFileListVec;
    private java.util.Vector proceduralSettingsElementListVec;
    private java.util.Vector userListVec;
}
```

APPENDIX C: TEST PROCEDURE

A. PHASE ONE TEST PROCEDURE

CyberCIEGE: Scenario Definition Tool
Interface Test Check List

Reusable Set:

Function: File->New

Instructions: Create three new instances of this reusable set one named empty, one named partial and one named full. Leave all fields blank for the instance named empty, enter some data in a few of the fields for partial (what data to enter and in which fields is left to the judgment of the tester) and fill in all fields for the instance named full. Note any fields that cannot be populated.

	PASS	FAIL	N/A
Result:			
The form listeners continued to function			
No other errors occurred			
Tester Comments:			

Function: Scenario Element Management Combo Box->New

Instructions: Create two new forms for each of the previously created reusable sets (for a total of three forms per set). As before, leave all fields blank for the instance named empty, enter some data in a few of the fields for partial and fill in all fields for the instance named full.

	PASS	FAIL	N/A
Result:			
The form listeners continued to function			
No other errors occurred			
Tester Comments:			

Function: Scenario Element Management Combo Box-> switching between forms in a set

Instructions: Progressively move from the first to the last elements in the set and then from the last back to the first.

Result:	PASS	FAIL	N/A
The form listeners continued to function			
No other errors occurred			

Tester Comments:

Function: File->Save

Instructions: Save each of the reusable sets

Result:	PASS	FAIL	N/A
The form listeners continued to function			
The file-system reflects the save			
The file is named correctly			
The reusable sets library tree reflects the save			
No other errors occurred			

Tester Comments:

Function: File->Save Scenario

Instructions: Save each of the reusable sets

Result:	PASS	FAIL	N/A
The form listeners continued to function			
The file-system reflects the save			
The file is named correctly			
The reusable sets library tree reflects the save			
No other errors occurred			

Tester Comments:

Function: File->Save As

Instructions: Save each of the reusable sets. Concatenate the number 1 to the file name

Result:	PASS	FAIL	N/A
The form listeners continued to function			
The file-system reflects the save			
The file is named correctly			

The reusable sets library tree reflects the save			
No other errors occurred			

Tester Comments:

Function: File->Save Scenario As

Instructions: Save each of the reusable sets. Concatenate the number 1 to the file name

Result:	**PASS**	**FAIL**	**N/A**
The form listeners continued to function			
The file-system reflects the save			
The file is named correctly			
The reusable sets library tree reflects the save			
No other errors occurred			

Tester Comments:

Function: Tabbed Work Area Right Click Menu -> Remove

Instructions: Remove each of the resuable set tabs

Result:	**PASS**	**FAIL**	**N/A**
No other errors occurred			

Tester Comments:

Function: File->Open

Instructions: Open the previously created reusable set named empty and empty1

Result:	**PASS**	**FAIL**	**N/A**
The form listeners continued to function			
No other errors occurred			

Tester Comments:

Function: Reusable Library Tree -> double click the file name to open

Instructions: Open the previously created reusable set named partial and partial1

	PASS	**FAIL**	**N/A**

Result:

The form listeners continued to function

No other errors occurred

Tester Comments:

Function: Reusable Library Tree Right Click Menu -> Open

Instructions: Open the previously created reusable set named full and full1

	PASS	FAIL	N/A

Result:

The form listeners continued to function

No other errors occurred

Tester Comments:

Function: Scenario Element Management Combo Box-> Delete Button

Instructions: Delete all of the elements in each of the empty, partial and full reusable sets. DO NOT delete the elements in the empty1, partial1 and full1 reusable sets.

	PASS	FAIL	N/A

Result:

No other errors occurred

Tester Comments:

Function: Reusable Library Tree Right Click Menu -> Delete

Instructions: Delete the empty, partial and full reusable sets. DO NOT delete the empty1, partial1 and full1 reusable sets.

	PASS	FAIL	N/A

Result:

No other errors occurred

Tester Comments:

Function: Reusable Library Tree Right Click Menu -> Add

Instructions: Add empty1 to the Scenario.

	PASS	FAIL	N/A

Result:

The form listeners continued to function

	PASS	FAIL	N/A
The scenario tree reflects the save			
No other errors occurred			

Tester Comments:

Function: Drag-and-Drop

Instructions: Add partial1 to the scenario.

Result:	PASS	FAIL	N/A
The form listeners continued to function			
The scenario tree reflects the save			
No other errors occurred			

Tester Comments:

Function: Tabbed Work Area Right Click Menu -> Add

Instructions: Add full1 to the scenario.

Result:	PASS	FAIL	N/A
The form listeners continued to function			
The scenario tree reflects the save			
No other errors occurred			

Tester Comments:

REMOVE THE TABS FOR EMPTY1, PARTIAL1 AND FULL1 BEFORE CONTINUING TO THE NEXT TEST.

Function: Scenario Tree Double Click to Open

Instructions: Open empty1.

Result:	PASS	FAIL	N/A
The form listeners continued to function			
No other errors occurred			

Tester Comments:

Function: Scenario Tree Right Click Menu -> Open

Instructions: Open partial 1.

Result:	PASS	FAIL	N/A
The form listeners continued to function			

No other errors occurred			

Tester Comments:

Function: Scenario Tree Right Click Menu -> Remove

Instructions: Remove empty1, partial1 and full1 from the scenario.

Result:	**PASS**	**FAIL**	**N/A**
No other errors occurred			

Tester Comments:

LIST OF REFERENCES

[Deitel 2002] Deitel, H & Deitel, P. (2002). *Java: How to Program fourth Ed*. New Jersey: Prentice Hall.

[Fisher 2003] Fisher, C., Chiricosta, T. & Witherspoon, T. (2003). *Software Testing Workshop*. Proceedings of the STC 45th Annual Conference, Anaheim, California. Retrieved (February 2003) from the World Wide Web: http://www.ocstc.org/ana_conf/pdf/tt7s.pdf

[Geary 1999] Geary, D. (1999). *Graphic Java 2, Volume II: Swing 3rd ed*. Palo Alto: Sun Microsystems Press.

[Horstmann 2002-1] Horstmann, C & Cornell, G. (2002). *Core Java 2, Volume I: Fundamentals*. Palo Alto: Sun Microsystems Press.

[Horstmann 2002-2] Horstmann, C & Cornell, G. (2002). *Core Java 2, Volume II: Advanced Features*. Palo Alto: Sun Microsystems Press.

[Irvine 2002] Irvine, C. & Thompson, M. (2002). *SimSecurity -- Can You Keep the Network Alive?* Naval Postgraduate School Center for Information Systems Security Studies and Research. Retrieved (September 2003) from the World Wide Web: http://cisr.nps.navy.mil/SimSecurity/web/SimSecurity.html

[Irvine1 2003] Irvine, C. & Thompson, M. (2003, June). *Teaching Objectives of a Simulation Game for Computer Security.* Proceedings of Informing Science and Information Technology Joint Conference, Pori, Finland.

[Irvine2 2003] Irvine, C. (2003, May). *The SimSecurity Information Assurance Virtual Laboratory*. Proceedings of IEEE Security & Privacy conference, Oakland, California.

[Pressman 2001] Pressman, R. (2001). *Software Engineering: A Practitioner's Approach 5th ed*. Boston: Mc Graw Hill.

[Rivermind 2002] Rivermind, Inc. & Naval Postgraduate School Center for Information Systems Security Studies and Research (2002). *CyberCIEGE: Scenario Format Template.*

[Saltzer 1975] Jerome, S. & Schroeder, M. (1975, September). *The Protection of Information in Computer System*. Proceedings of the IEEE, vol. 63, no. 9, pp. 1278-1308.

[Sun 2003] Hypertext (2003). *Java TM 2 Platform, Standard Edition, v 1.4.2 API Specification.* Sun Microsystems. Retrieved (September 2003) from the World Wide Web: http://java.sun.com/j2se/1.4.2/docs/api/

[Teo 2002] Tiat Lang, T. (2002). *Scenario Selection and Student Assessment Modules for CyberCIEGE*. Master of Science Thesis, Department of Computer Science, Naval Postgraduate School.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Ken Allen
   Rivermind
   Mountain View, CA

4. George Bieber
   OSD
   Washington, DC

5. RADM Joseph Burns
   Fort George Meade, MD

6. Bill Chinn
   Rivermind
   Mountain View, CA

7. Deborah Cooper
   DC Associates, LLC
   Roslyn, VA

8. CDR Daniel L. Currie
   PMW 161
   San Diego, CA

9. LCDR James Downey
   NAVSEA
   Washington, DC

10. Scott Gallardo
    Rivermind
    Mountain View, CA

11. Richard Hale
    DISA
    Falls Church, VA

12. LCDR Scott D. Heller
SPAWAR
San Diego, CA

13. Wiley Jones
OSD
Washington, DC

14. Russell Jones
N641
Arlington, VA

15. David Ladd
Microsoft Corporation
Redmond, WA

16. Dr. Carl Landwehr
National Science Foundation
Arlington, VA

17. Steve LaFountain
NSA
Fort Meade, MD

18. Dr. Greg Larson
IDA
Alexandria, VA

19. Ray A. Letteer
Head, Information Assurance, HQMC C4 Directorate
Washington, DC

20. Penny Lehtola
NSA
Fort Meade, MD
pennyl@nsa.gov

21. Gilman Louie
In-Q-Tel
Menlo Park, CA 94025

22. Ernest Lucier
Federal Aviation Administration
Washington, DC

23. CAPT Sheila McCoy
Headquarters U.S. Navy
Arlington, VA

24. Dr. Ernest McDuffie
National Science Foundation
Arlington, VA

25. Dr. Vic Maconachy
NSA
Fort Meade, MD

26. Doug Maughan
Department of Homeland Security
Washington, DC

27. John Mildner
SPAWAR
Charleston, SC

28. Dr. John Monastra
Aerospace Corporation
Chantilly, VA

29. Brian Morgan
Rivermind
Mountain View, CA

30. Marshall Potter
Federal Aviation Administration
Washington, DC

31. Dr. Roger R. Schell
Aesec
Pacific Grove, CA

32. Keith Schwalm
Good Harbor Consulting, LLC
Washington, DC

33. Dr. Ralph Wachter
ONR
Arlington, VA

34. David Wirth
N641
Arlington, VA

35. Daniel Wolf
NSA
Fort Meade, MD

36. CAPT Robert Zellmann
CNO Staff N614
Arlington, VA

37. Albert Wong
Naval Postgraduate School
Monterey, CA

38. Michael F. Thompson
Naval Postgraduate School
Monterey, CA

39. Dr. Cynthia E. Irvine
Naval Postgraduate School
Monterey, CA

40. Paul C. Clark
Naval Postgraduate School
Monterey, CA

41. Ann E. Rideout
SPAWAR
Charleston, SC

42. Kenneth Johns
Civilian, Naval Postgraduate School
Monterey, CA